(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－187527

(43)公開日　平成10年(1998)７月21日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　12/00	５７２	Ｇ０６Ｆ　12/00	５７２Ａ
9/46	３４０	9/46	３４０Ｆ

審査請求　未請求　請求項の数３　ＦＤ　（全 22 頁）

(21)出願番号　特願平9－187269

(22)出願日　平成９年(1997)６月30日

(31)優先権主張番号　０８／６７３１３０

(32)優先日　1996年７月１日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591064003
サン・マイクロシステムズ・インコーポレーテッド
SUN MICROSYSTEMS, INCORPORATED
アメリカ合衆国　94303　カリフォルニア州・パロ　アルト・サン　アントニオ　ロード・901

(72)発明者　エリック・イー・ハガーステン
アメリカ合衆国・94043・カリフォルニア州・パロ　アルト・コーク　オーク　ウェイ・3451

(74)代理人　弁理士　山川　政樹

最終頁に続く

(54)【発明の名称】　アクセス競合を防ぐ装置および方法

(57)【要約】　（修正有）

【課題】　複数の記憶データ・オブジェクトを有し、複数のスレッドを同時に動作させることができるコンピュータ・システム内でアクセス競合を防ぐ。

【解決手段】　複数の動的ロック構造要素を有する動的ロック構造を備え、また、第１の複数の記憶データ・オブジェクトのうちの第２の複数の記憶データ・オブジェクトをマッピング関数に従って複数の動的ロック構造要素のうちの第１の動的ロック構造要素にマップする。第１の動的ロック構造要素は、第３の複数の記憶データ・オブジェクトの識別表示を記憶するように構成される。第３の複数の記憶データ・オブジェクトは、アクセスされている第２の複数の記憶データ・オブジェクトのサブセットを表し、したがって記憶データ・オブジェクトに現在アクセスしているスレッド以外のスレッドは、動的ロック構造内に記憶されているその識別表示を有する記憶データ・オブジェクトにアクセスすることができない。

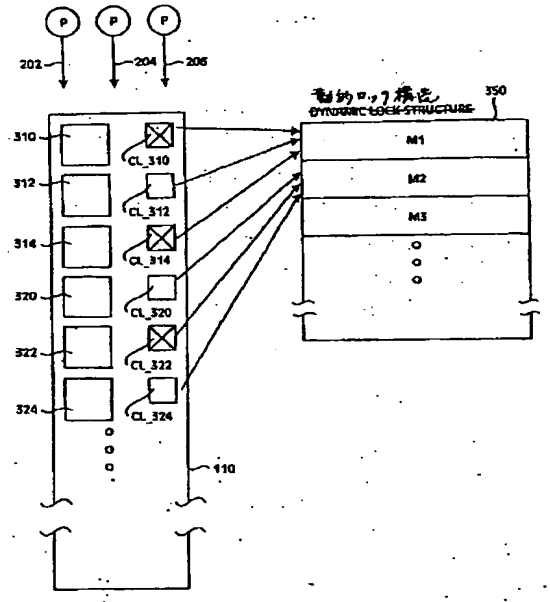


FIG. 3

1

【特許請求の範囲】

【請求項１】　第１の複数の記憶データ・オブジェクトを有し、複数のスレッドを同時に動作させることができるコンピュータ・システム内で、前記複数のスレッドのうちの２つ以上のスレッドが前記第１の複数の記憶データ・オブジェクトの１つに同時にアクセスすることを許可された場合に起こるアクセス競合を防ぐ装置において、

前記第１の複数の記憶データ・オブジェクトの数よりも数が少ない複数の動的ロック構造要素であって、前記第１の複数の記憶データ・オブジェクトのうちの第２の複数の記憶データ・オブジェクトが、マッピング関数に従って前記複数の動的ロック構造要素のうちの第１の動的ロック構造要素にマップし、前記マッピング関数が、前記第１の動的ロック構造要素にマップする前記第２の複数の記憶データ・オブジェクトのうちのただ１つの記憶データ・オブジェクトが前記複数のスレッドのうちの１つのスレッドによって所与の時点においてアクセスされるようにする、複数の動的ロック構造と、

現在アクセスされている前記第２の複数の記憶データ・オブジェクトのサブセットを表す第３の複数の記憶データ・オブジェクトの識別表示を記憶し、それにより現在前記記憶データ・オブジェクトにアクセスしているスレッド以外のスレッドが、前記動的ロック構造内に記憶されているその識別表示を有する記憶データ・オブジェクトにアクセスすることができない、前記第１の動的ロック構造要素に関連する記憶機能とを含んでいるアクセス競合を防ぐ装置。

【請求項２】　第１の複数の記憶データ・オブジェクトを有し、複数のスレッドを同時に動作させることができるコンピュータ・システム内で、前記複数のスレッドのうちの２つ以上のスレッドが前記第１の複数の記憶データ・オブジェクトの１つに同時にアクセスすることを許可された場合に起こるアクセス競合を防ぐ方法において、

複数の動的ロック構造要素を有する動的ロック構造を与えるステップと、

前記第１の複数の記憶データ・オブジェクトのうちの第２の複数の記憶データ・オブジェクトをマッピング関数に従って前記複数の動的ロック構造要素の第１の動的ロック構造要素にマップし、それにより前記複数の動的ロック構造要素が前記第１の複数の記憶データ・オブジェクトの数よりも数が少なくなる、マッピングするステップとを含み、前記第１の動的ロック構造要素が、アクセスされている前記第２の複数の記憶データ・オブジェクトのサブセットを表す第３の複数の記憶データ・オブジェクトの識別表示を記憶するように構成され、それにより現在前記記憶データ・オブジェクトにアクセスしているスレッド以外のスレッドが、前記動的ロック構造内に記憶されているその識別表示を有する記憶データ・オブジェクトにアクセスできないようにすることを特徴とする、アクセス競合を防ぐ方法。

2

【請求項３】　第１の複数の記憶データ・オブジェクトを有し、第１のスレッドと第２のスレッドを同時に動作させることができるコンピュータ・システム内で、前記第１のスレッドおよび第２のスレッドが前記第１の複数の記憶データ・オブジェクトのうちの第１の記憶データ・オブジェクトに同時にアクセスすることを許可された場合に起こるアクセス競合を防ぐ方法において、

前記第１のスレッドによって試みられた前記第１の記憶データ・オブジェクトへのアクセスを容易にするステップを含み、このアクセスを容易にするステップには、

前記動的ロック構造要素が記憶データ・オブジェクトの識別表示を記憶していない場合に、前記第１の記憶データ・オブジェクトの識別表示を前記動的ロック構造の物理ロック部分内に記憶するステップと、

前記動的ロック構造要素が前記第１の記憶データ・オブジェクトと異なる第２の記憶データ・オブジェクトのただ１つの識別表示を前記物理ロック部分内に記憶している場合に、前記第１の記憶データ・オブジェクトの前記識別表示ならびに前記第２の記憶データ・オブジェクトの前記識別表示を前記動的ロック構造の動的リスト部分内に記憶するステップ、および前記動的リストが現在空でないことを示す第１のフラグを前記動的ロック構造の前記物理ロック部分内に記憶するステップと、

前記動的ロック構造要素が前記第１の記憶データ・オブジェクトと同じ第２の記憶データ・オブジェクトのただ１つの識別表示を記憶している場合に、前記第１のスレッドによって試みられた前記第１の記憶データ・オブジェクトへの前記アクセスを失敗したものとして拒否するステップと、

前記物理ロックが、前記動的リストが現在空でないことを示す前記第１のフラグを記憶している場合に、前記第１の記憶データ・オブジェクトの前記識別表示がすでに前記動的リスト内に記憶されているかどうかを確認するために前記動的リストを調べるステップとを含んでおり、この動的リストを調べる前記ステップが、

前記動的リストが既に前記第１の記憶データ・オブジェクトの前記識別表示記憶している場合に、前記第１のスレッドによって試みられた前記第１の記憶データ・オブジェクトへの前記アクセスを失敗したものとして拒否するステップと、

前記動的リストがまだ前記第１の記憶データ・オブジェクトの前記識別表示記憶していない場合に、前記第１のスレッドによって試みられた前記第１の記憶データ・オブジェクトへの前記アクセスを許可し、前記第１の記憶データ・オブジェクトの前記識別表示を前記動的リスト内に記憶するステップとを含んでいることを特徴とする方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、コンピュータ・システム内の記憶データ・オブジェクトを共用する方法および装置に関し、さらに詳細には、本発明は、複数のプロセスまたはスレッドが複数の記憶されたデータ・オブジェクトに効率的かつ正確にアクセスすることができる方法および装置に関する。

【０００２】

【従来の技術】現代のコンピュータ・システムでは、複数のプロセスまたはスレッドが書込みまたは読取りを行うために同じ記憶データ・オブジェクトにアクセスしたいことがある。ここで使用するソフトウェア・プロセスまたはスレッドなる語は、同じ意味であり、一般に論理上別個の実行ストリームをさす。例として、同時に動作している複数の機能は、異なるプロセスまたはスレッド内で実行されていると言える。

【０００３】一般に、単一のプロセッサ、複数のプロセッサ、またはコンピュータ・ネットワーク内の複数のコンピュータを介して複数のスレッドを実行することができる。説明のために、図１に、複数のスレッドがプロセッサ１０２を介して多重方式で実行される多重プログラミング状況の一例を示す。スレッドから見ると、スレッドは、１つの多重プロセッサ内ではなく異なるプロセッサ内で実行されるように見える。

【０００４】図１には、図１のコンピュータ・システムに関連し、データを記憶する記憶スペースを示す記憶機能１１０が示されている。記憶機能（ＳＦ，storage facility）１１０は、例えば、半導体メモリ、フラッシュ・メモリ、仮想メモリ、磁気記憶装置または光記憶装置など、データを記憶することができる任意の装置または複数の装置の組合せを表す。

【０００５】記憶機能１１０内には、複数の記憶されたデータ・オブジェクト１１２、１１４および１１６が示されている。これらの記憶データ・オブジェクト１１２、１１４および１１６のそれぞれは、スレッド１０４、１０６または１０８の１つが書込みまたは読取りを行いたいデータ記憶装置のユニットを表す。各記憶データ・オブジェクトは、スレッドによって実施される特定のタスクに適合するように幾通りもの形で構成され、配置される。例として、スレッド１０４、１０６および１０８がデータベース内のデータを操作するプロセスを表す場合、記憶データ・オブジェクト１１２は、例えば、スレッド１０４～１０８のいずれかが書込みまたは読取りを行うことができるデータベースを表す。もちろん、任意の数の記憶データ・オブジェクトを記憶機能１１０内に準備することができる。データベース内には、例えば、スレッドがアクセスすることができるレコードを表す何百万個以上もの記憶データ・オブジェクトがある。

【０００６】また、スレッドによって操作すべきデータを記憶するユニットを表すのに「オブジェクト」なる語を使用しているが、現代の文献においてソフトウェアにおいて「オブジェクト指向」手法と通常呼ばれるものに本発明を限定する意図はないことを念頭に置かれたい。実際、本発明は、これに限定されないが、上述のオブジェクト指向手法を含む従来のプログラミング手法を使用して実施されるものである。

【０００７】上述のように、複数のスレッドは、複数のプロセッサを介して実施できる。図２および図３に、複数のスレッドが複数のプロセッサ（図２）上で実行される２つの異なる多重処理状況、および複数のコンピュータ・システム（図３）上で実行される２つの異なる多重処理状況を示す。図２には、それぞれ記憶データ・オブジェクト１１２、１１４および１１６にアクセスするスレッド１３０、１３２および１３４の別の１つを実行している複数のプロセッサ１２２、１２４および１２６を有するコンピュータ・システム１２０が示されている。図３には、スレッド１５０、１５２および１５４が異なるコンピュータ・システム１６０、１６２および１６４上で実行される別形態が示されている。各スレッド１５０、１５２および１５４は、記憶機能１１０の任意の記憶データ・オブジェクト１１２、１１４および１１６にアクセスする。

【０００８】２つ以上のスレッドが同じ記憶データ・オブジェクトに対してアクセスを試みることがなければ、アクセス競合は起こらない。複数のスレッドが同じ記憶データ・オブジェクト、例えば図２の記憶データ・オブジェクト１１２に同時にアクセスしたい場合、記憶データ・オブジェクト１１２の同時アクセスは、誤った結果をもたらす。説明のために、スレッド１３０が記憶データ・オブジェクト１１２内のデータを更新したい状況について考える。スレッド１３０が記憶データ・オブジェクト１１２の内容を更新している間、他のスレッド１３２がそれを読み取るために記憶データ・オブジェクト１１２にアクセスすることを許可された場合、アクセス競合が起こる。これは、スレッド１３２が記憶データ・オブジェクト１１２の一部更新されたコピーしか得ることができないためである。したがって、アクセス競合のために、スレッド１３２は誤ったデータを受け取ることになる。

【０００９】従来技術において、上述のアクセス競合問題を回避するいくつかの技法が提案されている。図４に、複数のスレッドが単一の記憶データ・オブジェクトに同時にアクセスすることを許可された場合に起こるアクセス競合を回避する従来技術の技法を示す。次に図４を参照すると、記憶機能１１０内の記憶データ・オブジェクト１８０、１８２、１８４、１８６、１８８および１８９にアクセスすることができるスレッドを表す複数のスレッド２０２、２０４および２０６が示されている。アクセス競合を回避するために、各記憶データ・オブジェクト１８０～１８９にはロッキング・フラグが関

連付けをされている。例えば、それぞれの記憶データ・オブジェクト１８０、１８２、１８４、１８６、１８８および１８９に対応するロッキング・フラグ１９０、１９２、１９４、１９６、１９８および１９９が示されている。

【００１０】スレッド、例えばスレッド２０２は、記憶データ・オブジェクト、例えば記憶データ・オブジェクト１８０にアクセスした場合、アクセスした記憶データ・オブジェクトに対応するロッキング・フラグをセットする。記憶データ・オブジェクト１８０の場合、スレッド２０２より記憶データ・オブジェクト１８０がアクセスされたとき、対応するロッキング・フラグ１９０がセットされる。ロッキング・フラグ１９０がセットされている間、他のスレッド、例えばスレッド２０４または２０６は、記憶データ・オブジェクト１８０にアクセスすることができない。スレッド２０２は、そのアクセスを終了すると、ロッキング・フラグ１９０をリセットし、それにより他のスレッドが記憶データ・オブジェクト１８０にアクセスすることができるようにする。

【００１１】アクセス競合を回避する従来技術の技法は、比較的少数の記憶データ・オブジェクトに対しては十分に役立つ。しかしながら、多数の記憶データ・オブジェクト、例えば何百万もの記憶データ・オブジェクトを有するシステムにおいては、すべての記憶データ・オブジェクトに対するフラグの要求は、多数の記憶データ・オブジェクトに対して何百万個以上ものフラグが必要となるので、記憶機能資源の極めて非能率的な使用を呈する。各フラグが、例えば、フラグの読取りまたは書込みの速度を高めるために通常行われるように、１バイト（すなわち８ビット）または１ワード（すなわち３２ビット）を必要とする場合、すべての従来技術のフラグを実施するのに必要なビットの数は法外に多くなる。各フラグが１ビットしか必要としない場合でも、そのような多数のフラグは、システムによっては、アクセス競合を防ぐためだけに容認できないほど大きい記憶機能オーバヘッドとなる。

【００１２】さらに、記憶データ・オブジェクト当たりのビットの数は、システムによってはあまり多くないこともある。この場合、実際のデータ記憶に使用されるビットの数に対するフラグを実施するのに必要なビットの数の比率は、容認できないほど大きくなる。このオーバヘッド率はまた、理解できるように、記憶データ・オブジェクトのサイズが縮小するにつれて増大する傾向がある。

【００１３】

【発明が解決しようとする課題】以上のことから、複数のスレッドが同じ記憶データ・オブジェクトにアクセスしようと試みた場合に起こるアクセス競合を回避する改善された方法および装置が必要である。

【００１４】

【課題を解決するための手段】本発明は、一実施形態では、第１の複数の記憶データ・オブジェクトを有し、複数のスレッドを同時に動作させることができるコンピュータ・システム内でアクセス競合を防ぐ装置に関する。アクセス競合は、複数のスレッドのうちの２つ以上のスレッドが第１の複数の記憶データ・オブジェクトの１つに同時にアクセスすることを許可された場合に起こる。

【００１５】本装置は、複数の動的ロック構造要素を有する動的ロック構造を含んでいる。複数の動的ロック構造要素は、第１の複数の記憶データ・オブジェクトの数よりも数が少ない。第１の複数の記憶データ・オブジェクトのうちの第２の複数の記憶データ・オブジェクトは、マッピング関数に従って複数の動的ロック構造要素のうちの第１の動的ロック構造要素にマップする。マッピング関数では、第１の動的ロック構造要素にマップする第２の複数の記憶データ・オブジェクトのうちのただ１つの記憶データ・オブジェクトが、複数のスレッドのうちの１つのスレッドによって所与の時点においてアクセスされるようになっている。

【００１６】また、第３の複数の記憶データ・オブジェクトの識別表示（アイデンティティ）を記憶する第１の動的ロック構造要素に関連付けをされた記憶機能が含まれている。第３の複数の記憶データ・オブジェクトは、現在アクセスされている第２の複数の記憶データ・オブジェクトのサブセットを表し、したがって記憶データ・オブジェクトに現在アクセスしているスレッド以外のスレッドは、動的ロック構造内に記憶されているその識別表示を有する記憶データ・オブジェクトにアクセスすることができない。

【００１７】他の実施形態では、第１の複数の記憶データ・オブジェクトを有し、複数のスレッドを同時に動作させることができるコンピュータ・システム内でアクセス競合を防ぐ方法に関する。この方法は、複数の動的ロック構造要素を有する動的ロック構造を与えるステップを含む。また、第１の複数の記憶データ・オブジェクトのうちの第２の複数の記憶データ・オブジェクトをマッピング関数に従って複数の動的ロック構造要素のうちの第１の動的ロック構造要素にマップするステップがある。マッピング関数のために、複数の動的ロック構造要素は、第１の複数の記憶データ・オブジェクトの数よりも数が少なくなる。第１の動的ロック構造要素は、第３の複数の記憶データ・オブジェクトの識別表示を記憶するように構成される。第３の複数の記憶データ・オブジェクトは、アクセスされている第２の複数の記憶データ・オブジェクトのサブセットを表し、したがって記憶データ・オブジェクトに現在アクセスしているスレッド以外のスレッドは、動的ロック構造内に記憶されているその識別表示を有する記憶データ・オブジェクトにアクセスすることができない。

【００１８】本発明の上記その他の利点は、以下の詳細

な説明を読み、各図面を検討すれば明らかになろう。

【００１９】

【発明の実施の形態】特に、複数のプロセスまたはスレッドが複数のデータ・オブジェクトに効率的かつ正確にアクセスすることができる本発明について説明する。以下の説明は、本発明を完全に理解することができるように、多数の特定の詳細を示す。しかしながら、本発明は、これらの特定の詳細の一部またはすべてを用いなくとも実施できることが当業者には明らかであろう。他の場合には、本発明を不必要に曖昧にしないために、周知の構造およびプロセス・ステップについては詳細に説明していない。

【００２０】本発明の一態様によれば、従来必要とされた全ての記憶データ・オブジェクトに対してのロッキング・フラグの要求をなくすせるので有利である。所与の時点においてアクセスされている記憶データ・オブジェクトの実際の数は、所与の記憶機能内の記憶データ・オブジェクトの総数よりも少ないことを理解されたい。この理解があれば、所与の時点において、どのスレッドからもアクセスされない記憶データ・オブジェクトが多数ある（したがって、アクセス競合の危険がない）ので、すべての記憶データ・オブジェクトに対して記憶機能内にロックを物理的に準備する必要はないということである。しかしながら、課題は、アクセス競合を防ぐために物理的に実施されるロックの数を減らす（それにより、この目的に充てられた記憶機能オーバヘッドを減らす）と同時に、スレッドから記憶データ・オブジェクトにアクセスすることができ、他のスレッドを迅速かつ効率的にロック・アウトすることである。

【００２１】本発明の一態様によれば、スレッドにより現在アクセスされている記憶データ・オブジェクトをロック・アウトする複数の物理ロックが提供される。本願で使用している物理ロックなる語は、一般にメモリ内で実際に実行されるデバイスをさし、その実行には、いくらかの記憶機能資源が必要である。本発明に従って提供される物理ロックの数は、図４の従来技術の手法と異なり、使用できる記憶データ・オブジェクトの数よりも数が少ないことが有利である。例えば、あるシステムでは、１０００個の物理ロックを使用して、何百万個の記憶データ・オブジェクトへのアクセスを追跡する何百万個の概念ロックを実施することもある。

【００２２】概念的に言えば、すべての記憶データ・オブジェクトには論理ロックすなわち概念ロックが関連付けられている。所与のシステムに対しての本発明の概念ロックの数は、従来技術の実際の物理ロックと同数であるが、本発明による概念ロックは、実行にあたり実際の記憶機能資源を必要としないことが有利である。この明細書でのそれらの説明は、本発明を当業者に周知の用語で示すものである。概念ロックは、実世界のコンピュータ・システム内で実行する必要はないことを常に念頭に置かれたい。

【００２３】本発明の特徴および利点の詳細な説明のために図５を参照する。図５には、複数の記憶データ・オブジェクト３１０、３１２、３１４、３２０、３２２および３２４を記憶する記憶機能１１０が示されている。特定の記憶データ・オブジェクトがスレッドによって現在アクセスされているかどうか（したがって、他のスレッドによって試みられたアクセスをロック・アウトすべきかどうか）を概念的に示すために、それぞれの記憶データ・オブジェクト３１０、３１２、３１４、３２０、３２２および３２４に関連付けをされた複数の概念ロックＣＬ＿３１０、ＣＬ＿３１２、ＣＬ＿３１４、ＣＬ＿３２０、ＣＬ＿３２２およびＣＬ＿３２４が示されている。例えば、概念ロックＣＬ＿３１０は、ロック状態にあり、スレッド、例えばスレッド２０２が記憶データ・オブジェクト３１０に現在アクセスしており、他のスレッド、例えばスレッド２０４または２０６によって試みられる記憶データ・オブジェクト３１０へのアクセスを拒否すべきことを示すものとして示されている。同様に、概念ロックＣＬ＿３１４は、ロック状態にあり、それにより現在その対応する記憶データ・オブジェクト３１４にアクセスしているスレッドを除いて、他のスレッドによってその関連付けをされた記憶データ・オブジェクト３１４へのアクセスを防ぐものとして示されている。一方、概念ロックＣＬ＿３１２、ＣＬ＿３２０、ＣＬ＿３２２およびＣＬ＿３２４は、非ロック状態にあり、スレッド２０２、２０４および２０６のいずれかがそれらに関連付けをされた記憶データ・オブジェクトへのアクセスを要求することができ、それらの要求は直ちに許可できることを示すものとして示されている。

【００２４】所与の時点においてロックされた概念ロックを追跡するために、本発明の一態様によれば、動的ロック構造３５０が提供される。動的ロック構造３５０の役目は、アクセス競合を防ぐために記憶データ・オブジェクトへのアクセスを管理することである。概念ロックと異なり、動的ロック構造３５０およびその構成部分は、実際には組み込まれたメモリである。すなわち、動的ロック構造３５０を実行する場合、実際には実際の物理記憶機能資源が使用される。動的ロック構造３５０は、スレッドによって現在アクセスされている概念ロックの識別表示（これは一実施形態において記憶データ・オブジェクトの識別表示を表す）のみを追跡することが必要であるので、その実施に充てられた記憶機能資源の総量は、従来技術において必要とされる量、例えば、従来技術図４のロック・フラグを実施するのに必要なメモリの量と比較して少ないことが有利である。

【００２５】図５に示すように、動的ロック構造３５０は複数の構造要素を含んでおり、そのうちの３つの要素Ｍ１、Ｍ２およびＭ３が示されている。本発明の一態様によれば、動的ロック構造３５０内の動的ロック構造要

素の数は、概念ロックの数よりも少ないことが有利である。動的ロック構造要素の数を少なくするために、複数の概念ロックが、何らかのマッピング関数に従って各動的ロック構造要素にマップされる。適切なマッピング関数により、同時に使用されている概念ロックが異なる動的ロック構造要素にマップされるようになり、同時に迅速に実行されることが好ましい。システムは、例えば、記憶データ・オブジェクトのアドレスの中央部から（例えば、適切なビット・マスクおよびシフト命令を使用して）ビット群を選択するか、あるいは従来のハッシュ関数を使用することができる。図５の例では、ＣＬ＿３１０からＣＬ＿３１９までのすべての概念ロックは動的ロック構造要素Ｍ１にマップし、ＣＬ＿３２０からＣＬ＿３２９までのすべての概念ロックは動的ロック構造要素Ｍ２にマップする、などとなる。

【００２６】図６および図７に、動的ロック構造要素Ｍ１、Ｍ２およびＭ３を含む動的ロック構造３５０の一部を本発明の一実施形態に従って詳細に示す。各動的ロック構造要素、例えば動的ロック構造要素Ｍ１内には、物理ロック部分、リストロック・フラグ部分および動的リスト部分が示されている。一実施形態では、物理ロック部分は、対応する動的構造要素にマップする１つの概念ロックの識別表示か、またはこの動的ロック構造要素に関連付けをされた動的リストが空かどうかを示すフラグを記憶するメモリ構造を含んでいる。概念ロックは概念としてのみ存在するので、概念ロックの識別表示は、一実施形態では、その対応する記憶データ・オブジェクトの識別表示によって表されることを理解されたい。記憶データ・オブジェクトは、例えば、メモリ内に記憶されている場合、その開始アドレスによって識別される。物理ロック部分内に記憶されるフラグＥＭＰＴＹ及びＮＯＴ＿ＥＭＰＴＹは、一実施形態では、単一ビット、例えば、単一ビットの０と１の値によって実行される。或いは、他の従来のロック実施形態を使用することもできる。

【００２７】図７の動的ロック構造要素Ｍ１を参照すると、例えば、物理ロック４１０は、動的ロック構造要素Ｍ１に関連付けをされた動的リスト４１２が空でないことを示すフラグＮＯＴ＿ＥＭＰＴＹを記憶するものとして示されている。一方、動的ロック構造要素Ｍ３の物理ロック４３０は、動的ロック構造要素Ｍ３の対応する動的リスト４３２が現在空であることを示すフラグＥＭＰＴＹを記憶するものとして示されている。

【００２８】動的ロック構造要素にマップされる概念ロックは、その動的ロック構造要素にマップするすべての概念ロックの中でスレッドによって現在アクセスされている唯一の概念ロックであるならば、その概念ロック識別表示が、その動的ロック構造要素の物理ロック部分内に記憶される。例として、動的ロック構造要素Ｍ２にマップするすべての概念ロックの中で概念ロックＣＬ＿３２２のみが現在アクセスされている場合、概念ロックＣＬ＿３２２の識別表示は、動的ロック構造要素Ｍ２の物理ロック部分、すなわち物理ロック部分４２０内に記憶されることが好ましい。後で詳細に説明するように、概念ロック識別表示が動的ロック構造要素の物理ロック部分内に記憶できる場合は常に、スレッドが記憶データ・オブジェクトにアクセスでき、本発明ではこのことを利用して速度を高め、概念ロックを獲得したり、解放することができる。

【００２９】動的ロック構造３５０の各動的ロック構造要素はさらにリストロック・フラグを含んでいる。リストロック・フラグは、それに関連付けをされた動的リスト、すなわち同じ動的ロック構造要素に属する動的リストが所与の時点においてアクセス可能であるかどうかを示すために使用される。図７の実施形態では、リストロック・フラグは、ＬＯＣＫ＿ＬＩＳＴおよびＵＮＬＯＣＫ＿ＬＩＳＴの２つの状態を有し、一実施形態では、単一ビット、例えば、ある単一ビットの０と１の値によって実行される。例として、動的ロック構造要素Ｍ１は、「ＬＯＣＫ＿ＬＩＳＴ」状態を現在記憶しており、対応する動的リスト４１２へのアクセスが現在許可されていないことを示すリストロック・フラグ４１４を有するものとして示されている。

【００３０】本発明の一態様によれば、各動的ロック構造要素の動的リスト部分は、概念ロック識別表示の動的に生成されたリストを含んでいる。動的リスト内に記憶されている概念ロック識別表示は、その動的リストに対応する動的ロック構造要素にマップし、それらに関連付けをされた記憶データ・オブジェクトがアクセスされているので現在「ロック」されている概念ロックの識別表示を表す。本発明の一態様によれば、動的に生成されたリスト、例えば、リンクされたリスト・データ構造を使用して概念ロック識別表示を記憶することにより、記憶機能の使用がフレキシブルかつ有利に最小限に抑えられる。

【００３１】動的リストは、例えば、それぞれ動的ロック構造要素Ｍ２およびＭ３の動的リスト４２２および４３２の場合に示されるように、空であり得る。動的ロック構造要素内の空の動的リストに関連付けられた物理ロック部分は、（動的ロック構造要素に現在１つの概念ロック識別表示しか記憶されていない場合）その関連付けられた動的ロック構造要素にマップされて現在ロックされている１つの概念ロックの識別表示を記憶するか、または、（動的リスト部分にも動的ロック構造要素の物理ロック部分にも概念ロック識別表示が記憶されていない場合）フラグＥＭＰＴＹを記憶する。概念ロック識別表示を記憶している物理ロックおよびＥＭＰＴＹフラグを記憶している物理ロックは、動的ロック構造要素Ｍ２の物理ロック４２０および動的ロック構造要素Ｍ３の物理ロック４３０に示されている。

【００３２】或いは、動的リストは、複数の概念ロック識別表示を含んでいることもある。上述のように、動的ロック構造要素が１つの概念ロック識別表示しか記憶していない場合、その概念ロック識別表示は、アクセス速度を高めるために、動的ロック構造要素の物理ロック部分内に記憶されることが好ましい。一方、動的ロック構造要素が複数の概念ロック識別表示、すなわちこの動的ロック構造要素に何れもマップされてあるスレッドによってアクセスされている複数の概念ロックの識別表示を記憶している場合、これら複数の概念ロック識別表示はすべて動的リスト部分内に記憶されることが好ましい。したがって、動的リスト部分は、２つ以上の概念ロック識別表示を含んでいるか、または概念ロックの識別表示を１つも含んでいないかのどちらかである。動的ロック構造要素の動的リスト部分が空でない場合、その動的ロック構造要素に関連付けられた物理ロック部分は、ＮＯＴ＿ＥＭＰＴＹフラグを記憶していることが好ましい。図７を参照すると、ＮＯＴ＿ＥＭＰＴＹフラグは、あるスレッドによってアクセスされ、動的ロック構造要素Ｍ１にマップしている２つの記憶データ・オブジェクトに対応する２つの概念ロック識別表示を記憶するために動的リスト４１２が使用される場合は、動的ロック構造要素Ｍ１の物理ロック４１０部分内に記憶されるものとして示されている。

【００３３】動的ロック構造要素の物理部分内にフラグＮＯＴ＿ＥＭＰＴＹが存在する場合は、スレッドには、概念ロックを獲得したり解放をしようとする場合に、動的リストを調べるべき指示が与えられると有利である。一方、動的ロック構造要素の物理部分内にフラグＥＭＰＴＹが存在する場合、スレッドには、この動的ロック構造要素にマップする概念ロックのいずれも現在アクセスされていないこと、すなわちそれらのいずれでも獲得できることが指示されると有利である。動的ロック構造要素の物理部分内にある概念ロック識別表示が存在する場合、この動的ロック構造要素にマップするすべての概念ロックのうち、この概念ロック識別表示のみが現在保持されていること、そして、もし概念ロック識別表示が重要でなければ、動的リストが空となるだろう（したがって、この動的ロック構造にマップする他のすべての概念ロックが必要ならば獲得できる）から動的リストを調べる必要はないことを、スレッドに指示すると有利である。したがって（同時に保持されている概念ロックが異なる動的ロック構造要素にマップする限り、すなわち適切に選択したハッシュ関数によってもたらされ得る状況における限り）物理ロック部分のみを使用して、概念ロックを容易に獲得したり、解放することができる。

【００３４】本発明の動的ロック構造の動作および本発明のアクセス競合回避技法におけるその役割は、図８、図９、図１０および図１１を参照すればよりよく理解できよう。図８および図１０は、本発明の一態様によれば、スレッド、例えば図５のスレッド２０２が記憶データ・オブジェクトにアクセスしたい場合にアクセス競合が起こらないようにするためにとられるステップを示す。図９および図１１は、それぞれ図８および図１０のステップの説明をさらに助ける例を示す。

【００３５】図８は、本発明の一態様によれば、所与のスレッドが記憶データ・オブジェクトに関連付けられた概念ロックを獲得したい場合に、その概念ロックが保持されている間、他のスレッドがその記憶データ・オブジェクトにアクセスするのを防ぐために行うステップを示す。ステップ５１０において、概念ロックＸが保持されている間、他のスレッドが概念ロックＸに関連付けられた記憶データ・オブジェクトにアクセスするのを防ぐために、所与のスレッド、例えばスレッドＡは、所与の概念ロック、例えば概念ロックＸを獲得することを望む。

【００３６】概念ロックＸがマップされる動的ロック構造要素の動的リストが空であり、かつ関連付けられた物理ロックが他の概念ロック識別表示を記憶するのに使用されていない場合、概念ロックＸの識別表示をこの動的ロック構造要素の物理ロック部分内に記憶する（ステップ５１２）。概念ロックＸの識別表示の物理ロック部分内への記憶が成功した場合には、ステップ５１４に進み、そこで概念ロックＸが獲得され、スレッドＡが概念ロックＸを保持している間、他のスレッドが概念ロックＸに関連付けられた記憶データ・オブジェクトにアクセスできないことを示すメッセージを、概念ロックＸを獲得したいスレッド、例えばスレッドＡに戻す。

【００３７】一方、関連付けられた動的リストが空でない（したがって、関連付けられた物理ロックがＮＯＴ＿ＥＭＰＴＹフラグを記憶している）場合、または関連付けられた物理ロックが他の概念ロックの識別表示を記憶するのに使用されている場合には、ステップ５１６に進み、そこで関連付けられた動的リストをロック・アウトし、スレッドＡが関連付けられた動的リストの操作を開始できるようにする。概念ロックＸがマップされる動的ロック構造要素が１つまたは複数の概念ロック識別表示を現在保持している場合にのみ、ステップ５１６に進むことに留意されたい。他のスレッドが関連付けられた動的リストを現在ロック・アウトしている場合は、ステップ５１６において、関連付けられた動的リストを得られ、それをロック・アウトするまで待ち、その後図８の後続のステップに進むことが好ましい。

【００３８】ステップ５１７において、物理ロック内に保持されている概念ロック識別表示があればそれを一次記憶変数ＴＥＭＰ内に保存する。さらに、物理ロックの内容を、この動的ロック構造に関連付けられた動的リストが図８のステップの終了時に空でないことを示すフラグＮＯＴ＿ＥＭＰＴＹにセットする。ステップ５１７では、スレッドＡが概念ロックＸを獲得できる場合、物理ロック部分内に保持された概念ロック識別表示の動的リ

スト内への後続の記憶を容易にすると有利である。図５において後で理解できるように、概念ロックＸがすでに他のスレッドによって保持されており、スレッドＡがアクセスすることができない場合は、後続のステップにおいてステップ５１７を逆転させ、物理ロック内に記憶された値をも元に戻し、その後ロックＸを獲得する試みが失敗したことをスレッドＡに知らせる。

【００３９】動的リストの検査に進む前に、ステップ５４０において、フラグＥＭＰＴＹを含んでいるかどうかを確認するために位置ＴＥＭＰを検査する。例えば、スレッドＡがステップ５１２を終了した後でスレッドＡがステップ５１６を実行する前に、他のスレッドが動的ロック構造要素の物理ロック部分内に記憶されている最後の概念ロックをロック解除している場合、プロセス中のこの点においては、ＴＥＭＰ位置には、フラグＥＭＰＴＹが含まれている。

【００４０】位置ＴＥＭＰがフラグＥＭＰＴＹを含んでいる（とステップ５４０において確認された）場合は、ステップ５４２に進み、概念ロックＸの識別表示を物理ロック部分内に記憶する。その後で、ステップ５４４に進み、そこでステップ５１６のアンドゥ（解除）を行う、すなわち他のスレッドがこの動的リストにアクセスできるように関連付けられた動的リストをロック解除する。ステップ５１４において、概念ロックＸを得ることができ、スレッドＡが概念ロックＸを保持している間、他のスレッドが概念ロックＸに関連付けられた記憶データ・オブジェクトにアクセスできないことを示すメッセージを、概念ロックＸを獲得したいスレッド、例えばスレッドＡに戻す。理解できるように、ステップ５４０、５４２、５４４および５１４を通るパスは、スレッドＡが概念ロックＸを獲得できるようにするための関連付けられた動的リストの操作や検査が不要なので最適化されている。

【００４１】位置ＴＥＭＰがフラグＥＭＰＴＹを含んでいない（とステップ５４０において確認された）場合、ステップ５１８に進み、概念ロックＸが使用できるかどうかを確認するために、動的リストならびにこの動的ロック構造要素に関連付けられた位置ＴＥＭＰを検査する。概念ロックＸは、他のスレッドによって現在アクセスされており、したがってその識別表示がすでに関連付けられた動的リスト内または（スワップ・ステップ５１７において物理ロックからその値を得る）ＴＥＭＰ位置に記憶されている場合には、使用できない。使用できない場合、ステップ５１９に進み、そこで前に物理ロック部分内に記憶された値をも物理ロック部分に戻すことによって前のステップ５１７をアンドゥ（解除）する。次いで、ステップ５２０に進み、他のスレッドが関連付けられた動的リストにアクセスできるよう、関連付けられた動的リストをロック解除する。その後、ステップ５２２において、概念ロックＸを獲得する試みが失敗したことを知らせるメッセージをスレッドＡに戻す。

【００４２】一方、ステップ５１８において、概念ロックＸが使用できる（概念ロックＸの識別表示が関連付けられた動的リスト内にもＴＥＭＰ位置にも記憶されていない）ことが確認された場合、ステップ５２４に進み、そこで概念ロックＸの識別表示を含めて、この動的ロック構造にマップし、スレッドによって現在保持されているすべての概念ロックＩＤを、関連付けられた動的リスト内に記憶する。これで、ステップ５２４を実行した後、概念ロックＸに関連付けられた動的ロック構造にマップし、現在保持されている複数の概念ロック識別表示が関連付けられた動的リスト内に記憶されていることになる。

【００４３】次いで、ステップ５２８において、他のスレッドが関連付けられた動的リストにアクセスできるよう、この関連付けられた動的リストをロック解除する。次いで、ステップ５３０において、概念ロックＸを獲得する試みが成功したことを知らせるメッセージをスレッドＡに戻し、これで概念ロックが保持され、したがって他のスレッドはそれにアクセスすることができない。

【００４４】図９は、一例において、スレッド２０２が図７の概念ロックＣＬ＿３２２を獲得したいときの図８のステップを示す（ステップ５５０）。概念ロックＣＬ＿３２２は動的ロック構造要素Ｍ２にマップするので、ステップ５２２において、物理ロック４２０の値を（一実施形態では値０によって表される）フラグＥＭＰＴＹと比較し、物理ロック４２０の内容がＥＭＰＴＹに等しければ、概念ロックＣＬ＿３２２の識別表示を物理ロック４２０の内容とスワップする。

【００４５】概念ロックＣＬ＿３２２がマップする動的ロック構造要素、すなわち動的ロック構造要素Ｍ２が、その関連付けられた概念ロック、例えば図５の概念ロックＣＬ＿３２０、ＣＬ＿３２２およびＣＬ＿３２４の識別表示を記憶していない場合は、物理ロック４２０の内容は、ＥＭＰＴＹに等しくなる。物理ロック４２０がＥＭＰＴＹフラグを記憶していることがステップ５２２において確認された場合は、比較およびスワップ操作は成功であり、ステップ５５３に進み、そこで概念ロックＣＬ＿３２２の獲得の試みが成功したことをスレッド２０２に知らせる。ステップ５３３の後、概念ロックＣＬ＿３２２の識別表示を物理ロック４２０内に記憶する。

【００４６】一方、ステップ５２２において、物理ロック４２０の内容がフラグＥＭＰＴＹに等しくない（物理ロック４２０が動的ロック構造要素Ｍ２に関連付けられた動的リスト、すなわち動的リスト４２２が空でないことを示すフラグ状態ＮＯＴ＿ＥＭＰＴＹを記憶しているか、または物理ロック４２０が概念ロックの識別表示を現在記憶している）ことが確認された場合、ステップ５５４に進み、そこでリストロック・フラグ４２４を状態ＬＯＣＫ＿ＬＩＳＴにセットする。ステップ５５４にお

いて、動的リスト４２２にアクセスしようと試みる他のスレッドをロック・アウトすることによって、スレッド２０２のために動的リスト４２２を獲得する。一実施形態では、ＬＯＣＫ＿ＬＩＳＴは、単一ビットの値１だけで表される。

【００４７】リストロック・フラグ４２４をＬＯＣＫ＿ＬＩＳＴにセットすることにより、他のスレッドからの妨害なしにスレッド２０２のために動的ロック構造Ｍ２に関連付けられた動的リストを操作することができる。ステップ５５６において、一次記憶位置ＴＥＭＰをフラグＮＯＴ＿ＥＭＰＴＹにセットし、この一時記憶位置ＴＥＭＰを物理ロック４２０にスワップする。

【００４８】ステップ５５８において、概念ロックＣＬ＿３２２の識別表示がすでに動的リスト４２２内かまたは一次位置ＴＥＭＰ内に記憶されているかを確認する。概念ロックＣＬ＿３２２の識別表示がすでに動的リスト４２２内かまたは一次位置ＴＥＭＰ内に記憶されている場合、他のスレッドがすでに概念ロックＣＬ＿３２２を獲得している。この場、概念ロックＣＬ＿３２２を獲得するスレッド２０２による試みは失敗であり、ステップ５６０に進み、一次位置ＴＥＭＰの識別表示を物理ロック４２０内に戻す、即ち前のスワップ・ステップ５５６を解除する。

【００４９】ステップ５６２において、リストロック・フラグ４２４をＵＮＬＯＣＫ＿ＬＩＳＴにセットする、すなわち他のスレッドが動的リスト４２２にアクセスすることができるよう、前のロッキング・ステップ５５４を解除する。ステップ５６４において、概念ロックＣＬ＿３２２を獲得する試みが失敗したことを知らせるメッセージをスレッド２０２に戻す。

【００５０】一方、ステップ５５８において、所望の概念ロック、すなわち概念ロックＣＬ＿３２２の識別表示が動的リスト４２２内かまたは一次記憶位置ＴＥＭＰ内に記憶されていないことが確認された場合、概念ロックＣＬ＿３２２は、他のスレッドによって獲得されておらず、したがってスレッド２０２が獲得することができる。この場合は、ステップ５６６に進み、動的リスト４２２に概念ロックＣＬ＿３２２の識別表示を追加する。

【００５１】ステップ５７０において、以前のロッキング・ステップ５５４を解除するために、リストロック・フラグ４２４を状態ＵＮＬＯＣＫ＿ＬＩＳＴにセットし、それにより他のスレッドが動的リスト４２２にアクセスすることができるようにする。ステップ５７２において、概念ロックＣＬ＿３２２を獲得するスレッド２０２の試みが成功したことを知らせるメッセージをスレッド２０２に戻す。

【００５２】ステップ５８０において、位置ＴＥＭＰがＥＭＰＴＹに等しいことかどうかを検査する。位置ＴＥＭＰがＥＭＰＴＹに等しければ、ステップ５８２、５８４および５５３を含む高速パスを介して概念ロックＣＬ＿３２２の獲得を開始する。ステップ５８２～５８４は、前に図８に関して説明したステップ５４２～５４４に類似している。その後、ステップ５５３に進み、概念ロックＣＬ＿３２２を獲得する試みが成功したことをスレッド２０２に知らせる。

【００５３】図９において、スレッド２０２は、異なる３つのパスを介して概念ロックＣＬ＿３２２を獲得できることに留意されたい。動的ロック構造要素Ｍ２が前に概念ロック識別表示を記憶しておらず、かつ概念ロックＣＬ＿３２２がこの動的ロック構造要素Ｍ２によって記憶された第１の概念ロックを表す場合、ステップ５５０、５５２および５５３に進み、スレッド２０２が概念ロック３２２に迅速にアクセスすることができるようにする。これらのステップは、数が少ないだけでなく、ステップ５５２におけるスワップ操作が一般にプロセッサによって実行されうる高速操作の１つであるので比較的迅速である。この場合、本発明のアクセス競合回避システムの性能は、すべての単一記憶データ・オブジェクトのロック・フラグを記憶するために相当量の記憶機能オーバヘッドを必要とする従来技術の方法とほとんど同じくらい効率的である。

【００５４】一方、ステップ５５２とステップ５５４の間で競合状態が起こっている場合、関連付けられた動的リストの操作または検索を必要とせずに、比較的速いパス（５５０／５５２／５５４／５５６／５８０／３８２／５８４／４４３）を介して概念ロックの獲得を達成することができる。

【００５５】一方、動的ロック構造要素Ｍ２が既に１つまたは複数の概念ロック識別表示、例えば概念ロックＣＬ＿３２０、ＣＬ＿３２２およびＣＬ＿３２４の１つを保持している場合、ステップ５５０、５５２、５５４、５５６、５５８、５６６、５６８、５７０および５７２を含むパスに進み、スレッド２０２が概念ロックＸを獲得することができるようにし、またこれら複数の概念ロック識別表示を動的ロック構造要素Ｍ２の動的リスト内に記憶することができるようにする。

【００５６】図１０に、本発明の一態様に従って、スレッド、例えばスレッドＡが、前に獲得した概念ロック、例えば概念ロックＸを解放したい場合に行うステップ（ステップ６０２）を示す。ステップ６０４において、関連付けられた物理ロックの内容が概念ロックＸの識別表示を含んでいる場合、ＥＭＰＴＹフラグを関連付けられた物理ロックの内容内に記憶する。関連付けられた物理ロックの内容が概念ロックＸの識別表示である場合、他の概念ロックＩＤは、概念ロックＸがそれにマップする動的ロック構造要素によって記憶される。この場合は、ステップ６０４において、解放を迅速に行い、ステップ６０６において、概念ロックＸは解放され、他のスレッドが獲得のために使用できると考えられる。

【００５７】一方、関連付けられた物理ロックの内容が

概念ロックＸの識別表示ではない場合、概念ロックＸがマップする動的ロック構造要素は、複数の概念ロック識別表示を現在記憶している。この場合、ステップ６０８に進み、そこで、他のスレッドが現れて関連付けられた動的リストの修正をするのを防ぐために、この関連付けられた動的リストをロック・アウトすると同時に、スレッドＡのために、概念ロックＸを、関連付けられた動的リストからの除去によって解放する。

【００５８】ステップ６１０において再び、前のステップ６０４の検査およびスワップ操作を実施する。ステップ６１０の検査およびスワップ操作は、ステップ６０８において動的リストをロック・アウトしようとしている間、スレッドＡと現在同時に動作している他のスレッドは、概念ロックＸの識別表示を関連する物理ロック内に入れてしまっているので、（ステップ６０８において）関連付けられた動的リストをロック・アウトした後で行うことが望ましい。他のスレッドが概念ロックＸの識別表示を関連付けられた物理ロック内に入れる理由は、最適化と関係があり、図１０の残りの説明から明らかになろう。

【００５９】ステップ６１０において、関連付けられた物理ロックの内容が概念ロックＸの識別表示であり、検査およびスワップ操作が成功したことが確認された場合、ステップ６１２に進み、そこで他のスレッドが関連付けられた動的リストにアクセスすることができるように、関連付けられた動的リストをロック解除する、すなわちロック・アウト・ステップ６０８の逆を行う。ステップ６１２の実行をした後、概念ロックＸはスレッドＡによって解放されたと考えられ、他のスレッドが獲得することができる。

【００６０】一方、ステップ６１０において、関連付けられた物理ロックの内容がまだ概念ロックＸの識別表示でないこと、すなわち検査およびスワップ操作が再び失敗したことが確認された場合、ステップ６１４に進み、そこで概念ロックＸの識別表示を関連付けられた動的リストから除去する。概念ロックＸの識別表示は関連付けられた動的リスト内にはなく、また関連付けられた物理ロックの内容内になかった（とステップ６１０において判定された）ので、概念ロックＸがマップする動的ロック構造要素は、ステップ６１４を実行した後、概念ロックＸの識別表示を追跡しない。したがって、後続のスレッドが現れ、図１０のステップが終了した後、概念ロックＸを獲得することができる。

【００６１】ステップ６１６、６１８、６２０、６２２および６２４において、可能なら、概念ロックＸがマップする動的ロック構造要素がただ１つの概念ロック識別表示を記憶している場合、その概念ロック識別表示が好ましくは関連付けられた物理ロックの内容内に移動されるよう動的ロック構造要素を最適化する。従って、残りの１つの概念ロックＸを保持するスレッドは、それを解放したい場合、ステップ６０２、６０４および６０６を含むパスを介して迅速にその解放をすることができる。この最適化を行わなければ、フラグＮＯＴ＿ＥＭＰＴＹにセットされた物理ロックは、フラグＮＯＴ＿ＥＭＰＴＹにセットされない。ＥＭＰＴＹ状態に戻れば、概念ロックをパス６０２／６０４／６０６を介して迅速に解放することができるので有利である。

【００６２】ステップ６１６において、概念ロックＸの識別表示が（ステップ１６４において）関連付けられた動的リストから削除された後に、関連付けられた動的リストにただ１つの概念ロック識別表示が残されているかどうかを確認する。関連付けられた動的ロック構造要素が追跡すべき複数の概念ロックＩＤが残っている場合は、ステップ６２４に進み、そこでスレッドＡが概念ロックＸを解放するステップを終了する。

【００６３】一方、概念ロックＸがマップする動的ロック構造要素が、その関連付けられた動的リスト内にただ１つの概念ロック識別表示を追跡する場合は、ステップ６１６からステップ６１８へ進み、そこで関連付けられた物理ロックの内容を残りの１つの概念ロックの識別表示にセットする。ステップ６２０において、関連付けられた動的ロック構造要素がその物理ロック内ならびにその動的リスト内でこの残りの１つの概念ロックを追跡しないように、この残りの１つの概念ロックの識別表示を関連付けられた動的リストから削除する。ステップ６２２において、後続のスレッドが関連付けられた動的リストにアクセスすることができるように、関連付けられた動的リストをロック解除する、すなわちロッキング・ステップ６０８を逆転させる。ステップ６２４において、概念ロックＸを解放する場合に行うステップを終了する。

【００６４】図１０のステップは、（図７に鑑みて）図１１によって示される例を参照すればより完全に理解することができる。図１１において、スレッド２０２は、前に獲得した概念ロックＣＬ＿３２２を解放することを望む（ステップ６５２）。ステップ６５４において、物理ロック４２０の内容が概念ロックＣＬ＿３２２の識別表示を含んでいる場合、ＥＭＰＴＹフラグを（図７の）関連付けられた物理ロック４２０の内容内に記憶する。関連付けられた物理ロック４２０の内容が概念ロックＣＬ＿３２２の識別表示である場合、他の概念ロックＩＤは、概念ロックＣＬ＿３２２がマップする動的ロック構造要素、すなわち動的ロック構造要素Ｍ２によっては記憶されない。この場合、ステップ６５４において、解放を迅速に実施し、概念ロックＣＬ＿３２２が解放され、ステップ６５６において、他のスレッドが獲得のために使用できると考えられる。

【００６５】一方、関連付けられた物理ロック４２０の内容が概念ロックＣＬ＿３２２の識別表示を含んでいる場合、動的ロック構造要素Ｍ２は現在、複数の概念ロッ

ク識別表示（現在保持されている概念ロックＣＬ＿３２２の他に少なくとも２つ以上）を記憶していることが理解される。この場合、ステップ６５８に進み、そこで他のスレッドが現れて関連付けられた動的リスト４２２の修正をするのを防ぐために、この関連付けられた動的リスト４２２をロック・アウトすると同時に、スレッド２０２のために、概念ロックＣＬ＿３２２を、関連付けられた動的リスト４２２からの除去によって解放する。

【００６６】ステップ６６０において、は再び、前のステップ６５４の検査およびスワップ操作を実施する。ステップ６６０の検査およびスワップ操作は、ステップ６５８において関連付けられた動的リスト４２２をロック・アウトしようとしている間に、スレッド２０２と現在同時に動作している他のスレッドは、概念ロックＣＬ＿３２２の識別表示を関連付けられた物理ロック４２０内に入れてしまっているので、（ステップ６５８において）関連付けられた動的リスト４２２をロック・アウトした後で行うことが望ましい。他のスレッドが概念ロックＣＬ＿３２２の識別表示を関連付けられた物理ロック４２０内に入れる理由は、図１０のステップ６１６、６１８、６２０、６２２および６２４に関して前に論じた。

【００６７】図１１のステップ６６０において、関連付けられた物理ロック４２０の内容が概念ロックＣＬ＿３２２の識別表示であり、検査およびスワップ操作が成功したことが確認された場合、ステップ６６２に進み、そこで他のスレッドが関連付けられた動的リスト４２２にアクセスすることができるように、関連付けられた動的リスト４２２をロック解除する、すなわちロック・アウト・ステップ６５８を逆転させる。ステップ６６２を実行した後、概念ロックＣＬ＿３２２はスレッド２０２によって解放されたと考えられ、他のスレッドが獲得することができる。

【００６８】一方、図１１のステップ６６０において、関連付けられた物理ロック４２０の内容がまだ概念ロックＣＬ＿３２２の識別表示でないこと、すなわち検査およびスワップ操作が再び失敗したことが確認された場合、ステップ６６４に進み、そこで概念ロックＣＬ＿３２２の識別表示を関連付けられた動的リスト４２２から除去する。概念ロックＣＬ＿３２２の識別表示は関連付けられた動的リスト４２２内になく、また関連付けられた物理ロック４２０の内容内になかった（とステップ６６０において判定された）ので、動的ロック構造要素Ｍ２は、ステップ６６４を実施した後、概念ロックＣＬ＿３２２の識別表示を追跡しない。したがって、後続のスレッドが現れ、図１１のステップが終了した後、概念ロックＣＬ＿３２２を獲得することができる。

【００６９】ステップ６６６、６６８、６７０、６７２および６７４において、可能なら、動的ロック構造要素Ｍ２がただ１つの概念ロック識別表示を記憶している場合、概念ロック識別表示が好ましくは関連付けられた物理ロック４２０の内容内に移動されるよう、動的ロック構造要素Ｍ２を最適化する。したがって、残りの１つの概念ロックを保持するスレッドは、その概念ロックを解放したい場合、ステップ６５２、６５４および６５６を含むパスを介して迅速にそれを解放することができる。

【００７０】ステップ６６６において、関連付けられた動的リスト４２２が、概念ロックＣＬ＿３２２の識別表示が（ステップ６６４において）関連付けられた動的リスト４２２から削除された後に残っているただ１つの概念ロック識別表示を有しているかどうかを確認する。関連付けられた動的ロック構造要素Ｍ２により追跡されるべき複数の概念ロックＩＤが残っている場合、ステップ６７４に進み、そこでスレッド２０２が概念ロックＣＬ＿３２２を解放するステップを終了する。

【００７１】一方、動的ロック構造要素Ｍ２がその関連付けられた動的リスト４２２内にただ１つの概念ロック識別表示を追跡する場合、ステップ６６６からステップ６６８へ進み、そこで関連付けられた物理ロック４２０の内容を残りの１つの概念ロックの識別表示にセットする。ステップ６７０において、関連付けられた動的ロック構造要素Ｍ２がその物理ロック４２０内ならびにその動的リスト４２２内でこの残りの１つの概念ロックを追跡しないように、この残りの１つの概念ロックの識別表示を関連付けられた動的リストから削除する。

【００７２】ステップ６７２において、後続のスレッドが関連付けられた動的リスト４２２にアクセスすることができるように、関連付けられた動的リスト４２２をロック解除する、すなわちロックリスト・フラグ４２４をフラグ状態ＵＮＬＯＣＫ＿ＬＩＳＴにセットすることによってロッキング・ステップ６６８を逆転させる。

【００７３】本発明では、複数の動的リストを使用して、概念ロック識別表示を追跡することに留意されたい。さらに、その動的リストに関連付けられた動的ロック構造要素によって保持されている概念ロックが複数存在する場合、動的リストを使用する。代わりに単一の動的リストを使用して、保持されているすべての概念ロックを追跡した場合、スレッドは、所与のロックが現在保持されているかどうかを判定するために、またはそのリストから現在保持されているロックを削除するために、リスト全体を探査しなければならない。多数の概念ロックが保持されている場合、この探査には不当に時間がかかり、またその実施が実行不可能になる。

【００７４】本発明では、比較的少数の概念ロック（例えば、一実施形態では３つまたは４つ）を所与の動的ロック構造要素にマップする。したがって、動的リストによって保持される概念ロックの最大数は比較的小さくなり、したがって概念ロックの獲得または解放の速度が高くなる。さらに動的リストは、その動的リストに関連する動的ロック構造要素によって保持されている概念ロッ

クが複数存在する場合にのみ使用される。動的ロック構造要素がただ１つの概念ロック識別表示を追跡する場合、物理ロック部分を使用して追跡を行うことが好ましく、それにより、本発明は、すべての記憶データ・オブジェクトについてロックを物理的に必要とする従来技術とほとんど同じ速さで概念ロックの獲得または解放を行うことができるようになる。

【００７５】さらに、各動的ロック構造要素が平均してわずか数個または１つの概念ロックを保持するように動的ロック構造要素の最適数を選択することによって、コンピュータ・システムの性能を調整することができる。各動的ロック構造要素が平均してわずか１つの概念ロック識別表示を保持できるようにする（したがって物理ロックを最大限に使用する）ことによって性能を最適化する実施形態では、コンピュータ設計者は、所与の時点においてアクセスすべき記憶データ・オブジェクトの平均数にほぼ比例するように、動的ロック構造要素の数を選択することができる。この所与の時点においてアクセスすべき記憶データ・オブジェクトの平均数は、例えば、統計的方法または他の従来の方法によって試行錯誤して決定することができる。例えば、ある所与の時点において使用できる概念ロックの１０分の１しか保持されていないことが確認された場合、動的構造は、（Ｎ／１０）個の動的ロック構造要素またはその整数倍を有しており（ただしＮは、概念ロックの総数を表す）、したがって動的ロック構造要素内、すなわちその物理ロック部分内に平均してただ１つの概念ロック識別表示が記憶されることになる。したがって、アクセス速度が有利に最大になる。一実施形態では、設計者は、動的ロック構造要素の数を減らすことによって、速度とメモリ・オーバヘッドとの兼ね合いをはかることができる。動的ロック構造要素が平均して複数の概念ロック識別表示を保持している場合でもなお、従来技術のメモリ・オーバヘッドを減らす目的は有利に達成される。

【００７６】さらに、本発明は、本願に記載のロック以外のタイプのロックを使用するシステムにも十分同様に適用できる。あるシステムでは、例えば、スレッドが読取りまたは書込みのために記憶データ・オブジェクトにアクセスする場合、その記憶データ・オブジェクト上のロックが獲得される。他のシステムでは、複数のスレッドが記憶データ・オブジェクトを同時に読み取ることはできるが、その記憶データ・オブジェクトに書込みを行いたいスレッドがロックを獲得することはできないロックが存在する。本願に記載の発明を異なるタイプのロックに適合させることは、この開示を得た当業者の力量でできることである。

【００７７】以上、本発明についていくつかの好ましい実施形態に関して説明したが、本発明の範囲内に入る改変例、置換例および同等物もある。また、本発明の方法および装置を実施する他の方法も多数ある。したがって、首記の請求の範囲は、本発明の真の精神および範囲内に入るすべての改変例、置換例および同等物を含むものと解釈すべきものである。

【図面の簡単な説明】

【図１】複数のスレッドがプロセッサを介して多重方式で実行される多重プログラミング状況を示す図である。

【図２】複数のスレッドが複数のプロセッサ上で実行される２つの異なる多重処理状況を示す図である。

【図３】複数のスレッドが複数のコンピュータ上で実行される２つの異なる多重処理状況を示す図である。

【図４】複数のスレッドが単一の記憶データ・オブジェクトへのアクセスを許される場合に生じることあるアクセス競合を回避する従来の技術を示す図である。

【図５】本発明の一態様に従ってアクセス競合を回避するために使用される構造を示す図である。

【図６】本発明の一態様に従って図５の動的ロック構造を詳細に示す図である。

【図７】一例において図５の動的ロック構造の各要素中に存在する各種パラメータを示す図である。

【図８】一実施形態において記憶データ・オブジェクトにアクセスするために概念ロックを獲得したい場合に行うステップを示す流れ図である。

【図９】一実施形態において記憶データ・オブジェクトにアクセスするために概念ロックを獲得したい場合に行うステップを示す流れ図である。

【図１０】一実施形態において概念ロックを解放したい場合に行うステップを示す流れ図である。

【図１１】一実施形態において概念ロックを解放したい場合に行うステップを示す流れ図である。

【符号の説明】

２０２、２０４、２０６　スレッド

３１０、３１２、３１４、３２０、３２２、３２４　記憶データ・オブジェクト

３５０　動的ロック構造

【図1】

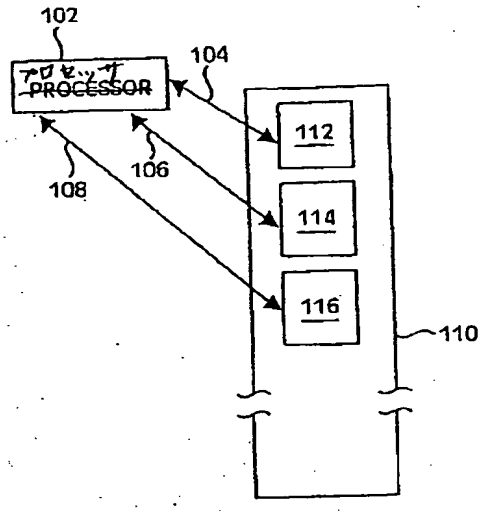


FIG. 1A

【図2】

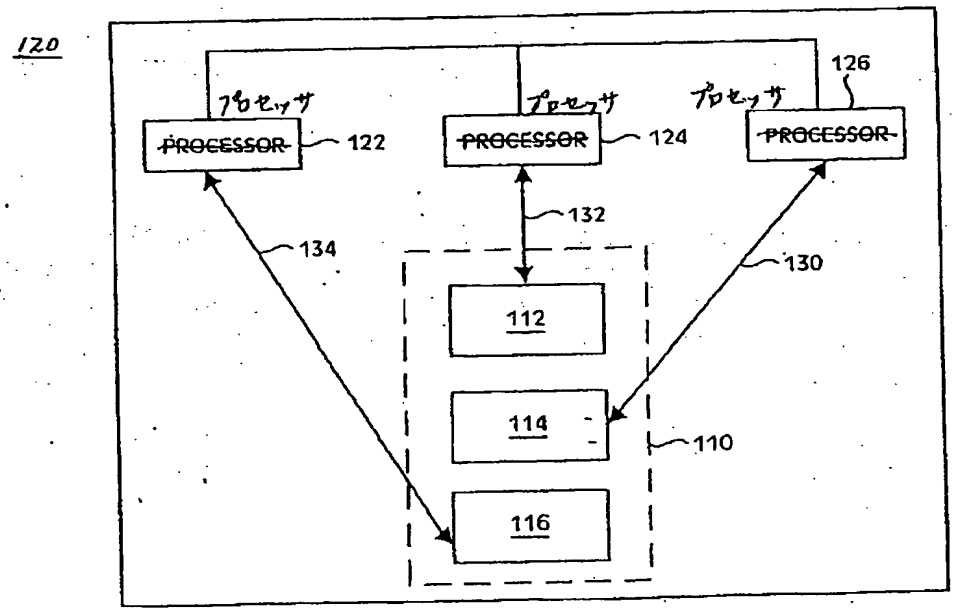


FIG. 1B

【図３】

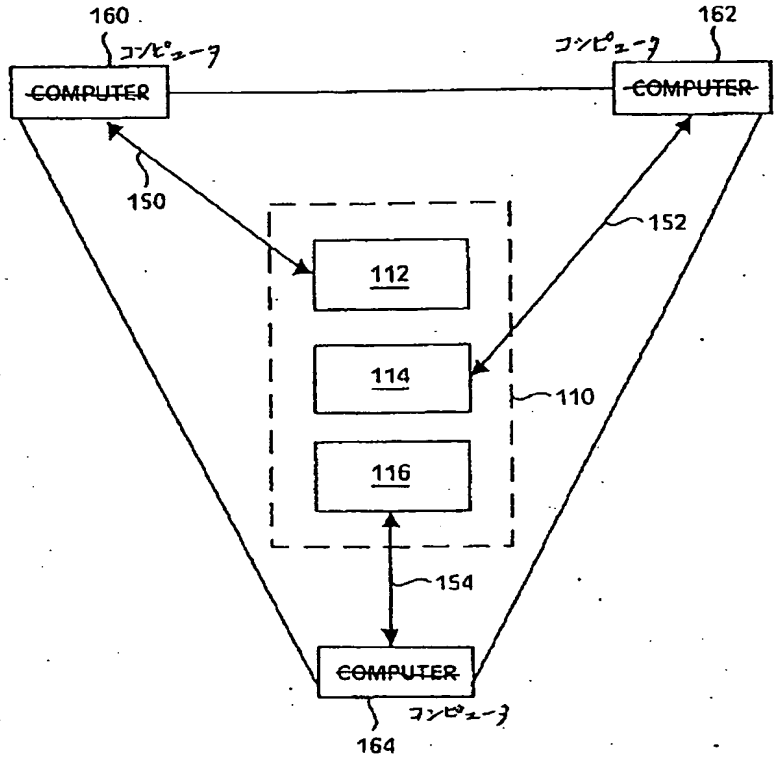

160
コンピュータ
COMPUTER
162
コンピュータ
COMPUTER
150
152
112
114
116
110
154
COMPUTER
コンピュータ
164

FIG. 1C

【図４】

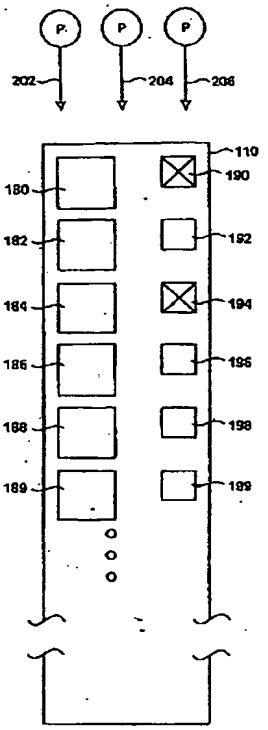


FIG. 2
(PRIOR ART)
/従来技術\

【図７】

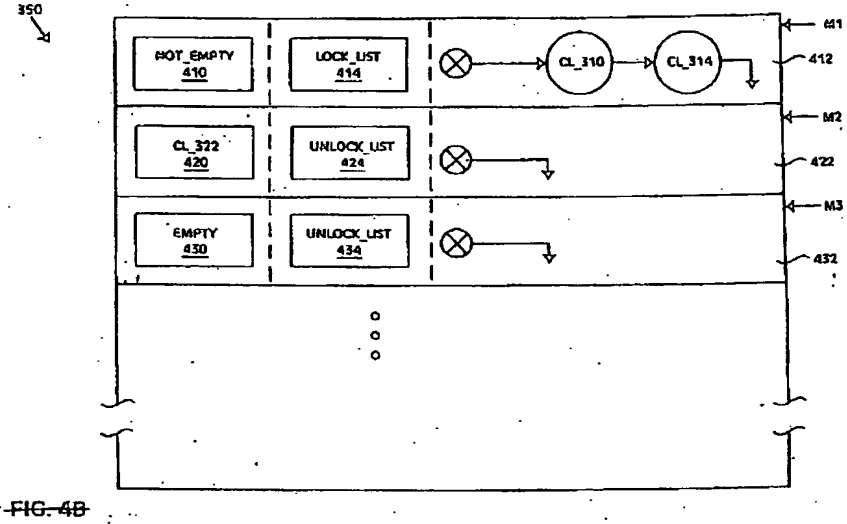


FIG. 4B

【図５】

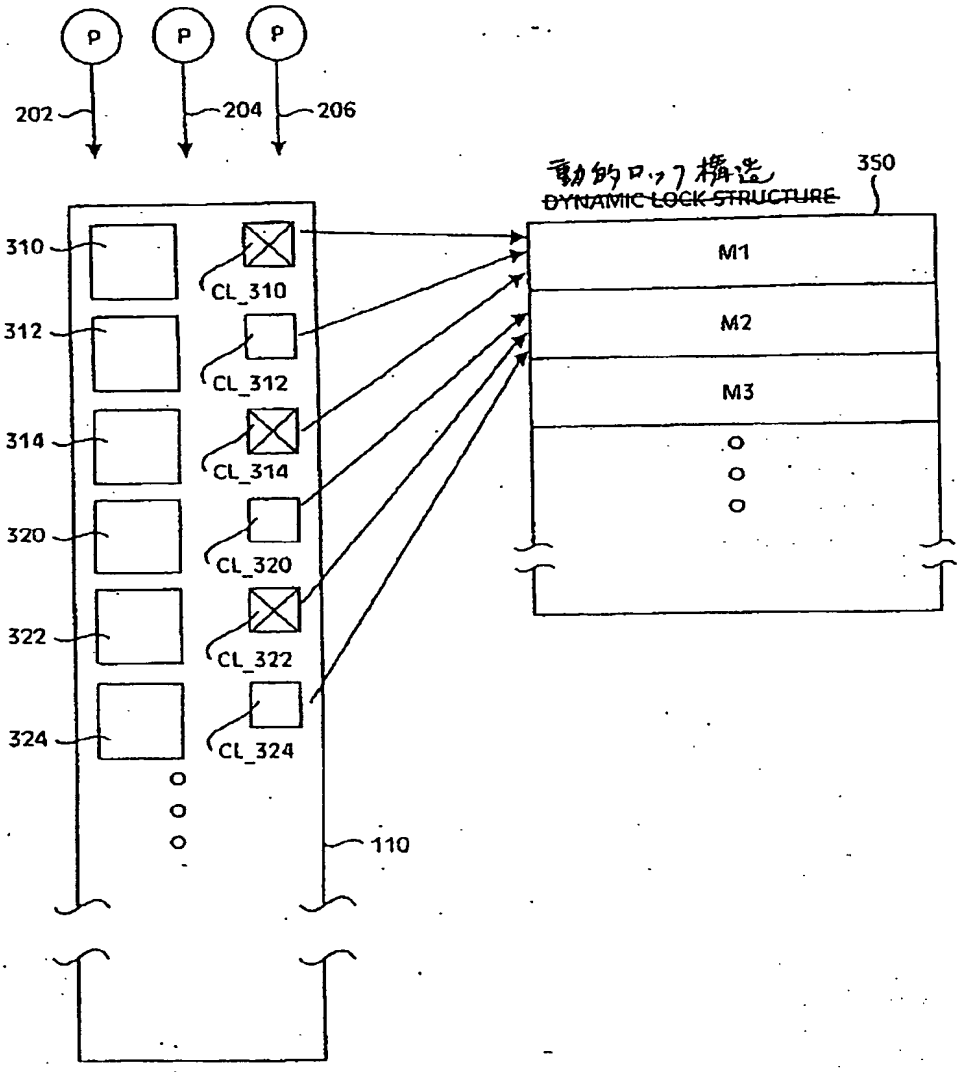

P
P
P
202
204
206
動的ロック構造
DYNAMIC LOCK STRUCTURE
350
M1
M2
M3
310
312
314
320
322
324
CL_310
CL_312
CL_314
CL_320
CL_322
CL_324
110

FIG. 3

【図6】

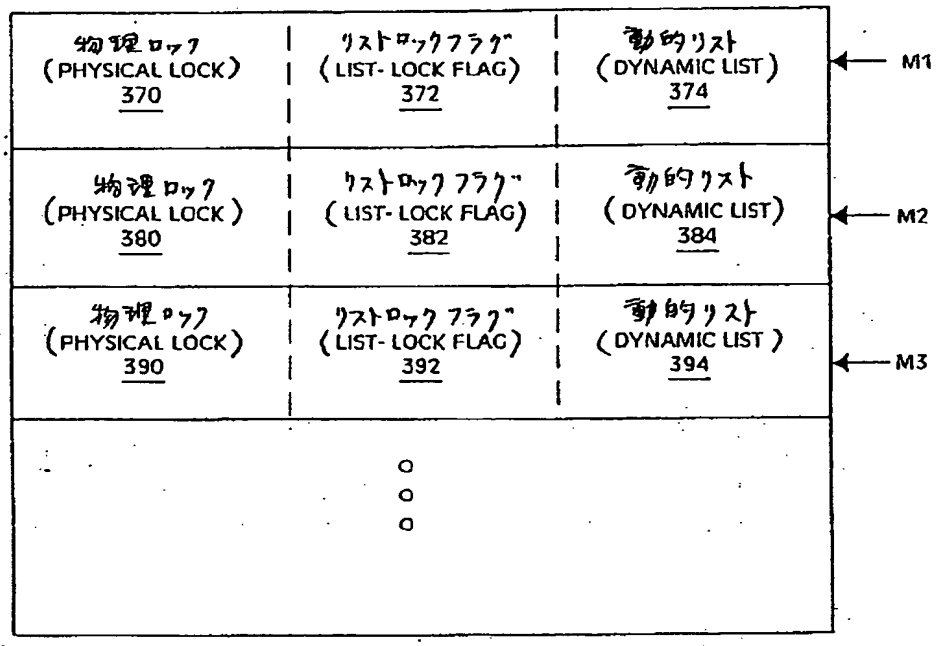

物理ロック
(PHYSICAL LOCK)
370
リストロックフラグ
(LIST-LOCK FLAG)
372
動的リスト
(DYNAMIC LIST)
374
M1
物理ロック
(PHYSICAL LOCK)
380
リストロックフラグ
(LIST-LOCK FLAG)
382
動的リスト
(DYNAMIC LIST)
384
M2
物理ロック
(PHYSICAL LOCK)
390
リストロックフラグ
(LIST-LOCK FLAG)
392
動的リスト
(DYNAMIC LIST)
394
M3
350

FIG. 4A

【図8】

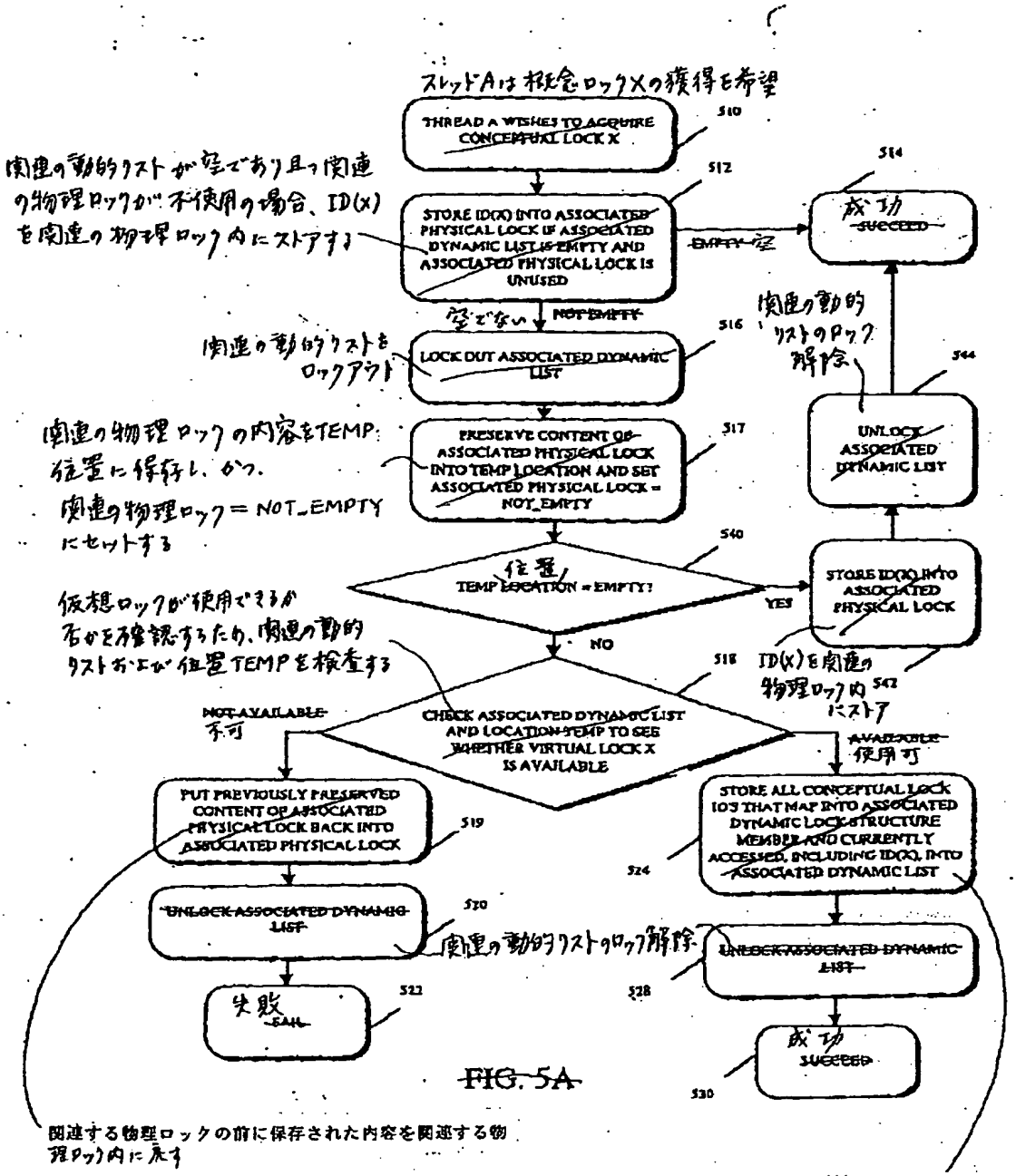

スレッドAは概念ロックXの獲得を希望
THREAD A WISHES TO ACQUIRE CONCEPTUAL LOCK X
510
関連の動的リストが空であり且つ関連の物理ロックが不使用の場合、ID(X)を関連の物理ロック内にストアする
STORE ID(X) INTO ASSOCIATED PHYSICAL LOCK IF ASSOCIATED DYNAMIC LIST IS EMPTY AND ASSOCIATED PHYSICAL LOCK IS UNUSED
512
EMPTY 空
514
成功
SUCCEED
空でない NOT EMPTY
関連の動的リストをロックアウト
LOCK OUT ASSOCIATED DYNAMIC LIST
516
関連の動的リストのロック解除
544
UNLOCK ASSOCIATED DYNAMIC LIST
関連の物理ロックの内容をTEMP位置に保存し、かつ、関連の物理ロック＝NOT_EMPTYにセットする
PRESERVE CONTENT OF ASSOCIATED PHYSICAL LOCK INTO TEMP LOCATION AND SET ASSOCIATED PHYSICAL LOCK = NOT_EMPTY
517
540
位置 TEMP LOCATION = EMPTY?
YES
STORE ID(X) INTO ASSOCIATED PHYSICAL LOCK
仮想ロックが使用できるか否かを確認するため、関連の動的リストおよび位置TEMPを検査する
NO
518
ID(X)を関連の物理ロック内にストア
542
CHECK ASSOCIATED DYNAMIC LIST AND LOCATION TEMP TO SEE WHETHER VIRTUAL LOCK X IS AVAILABLE
NOT AVAILABLE 不可
AVAILABLE 使用可
PUT PREVIOUSLY PRESERVED CONTENT OF ASSOCIATED PHYSICAL LOCK BACK INTO ASSOCIATED PHYSICAL LOCK
519
STORE ALL CONCEPTUAL LOCK ID'S THAT MAP INTO ASSOCIATED DYNAMIC LOCK STRUCTURE MEMBER AND CURRENTLY ACCESSED, INCLUDING ID(X), INTO ASSOCIATED DYNAMIC LIST
524
UNLOCK ASSOCIATED DYNAMIC LIST
520
関連の動的リストのロック解除
UNLOCK ASSOCIATED DYNAMIC LIST
528
失敗
FAIL
522
成功
SUCCEED
530
関連する物理ロックの前に保存された内容を関連する物理ロック内に戻す
ID（X）を含めて、関連する動的ロック構造要素にマップされ、現在アクセスされているすべての概念ロックIDを関連する動的リスト内に記憶する

FIG. 5A

【図9】

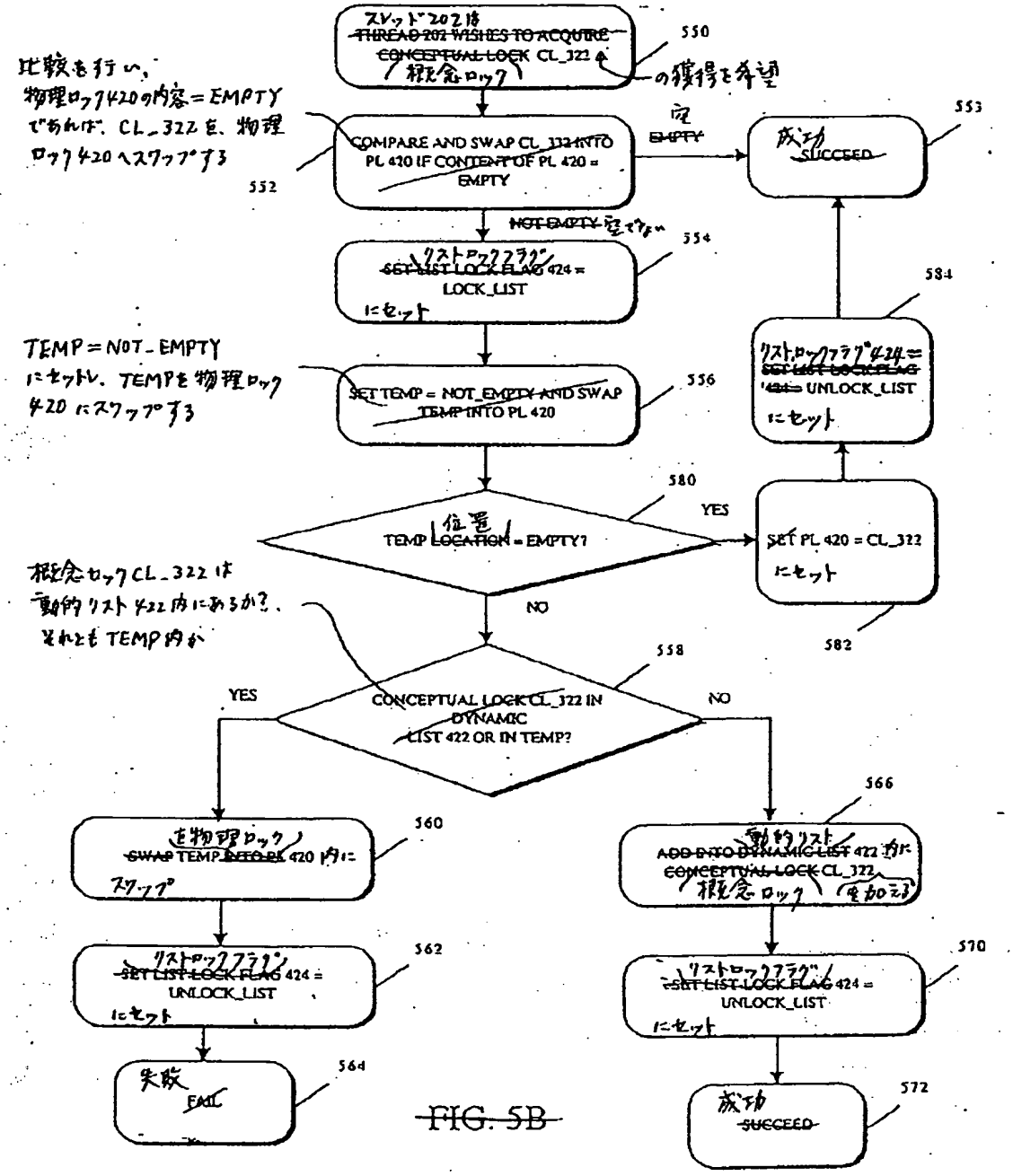

スレッド202は
THREAD 202 WISHES TO ACQUIRE
CONCEPTUAL LOCK CL_322
概念ロック
550
の獲得を希望
比較を行い、
物理ロック420の内容＝EMPTY
であれば、CL_322を、物理
ロック420へスワップする
COMPARE AND SWAP CL_322 INTO
PL 420 IF CONTENT OF PL 420 =
EMPTY
552
対
EMPTY
成功
SUCCEED
553
NOT EMPTY 空でない
リストロックフラグ
SET LIST LOCK FLAG 424 =
LOCK_LIST
にセット
554
リストロックフラグ424＝
UNLOCK_LIST
にセット
584
TEMP＝NOT_EMPTY
にセットし、TEMPを物理ロック
420にスワップする
SET TEMP = NOT_EMPTY AND SWAP
TEMP INTO PL 420
556
TEMP LOCATION = EMPTY?
580
YES
SET PL 420 = CL_322
にセット
582
概念ロックCL_322は
動的リスト422内にあるか、
それともTEMP内か
NO
CONCEPTUAL LOCK CL_322 IN
DYNAMIC
LIST 422 OR IN TEMP?
558
YES
NO
を物理ロック
SWAP TEMP INTO PL 420 内に
スワップ
560
動的リスト
ADD INTO DYNAMIC LIST 422 内に
CONCEPTUAL LOCK CL_322
概念ロック
を加える
566
リストロックフラグ
SET LIST LOCK FLAG 424 =
UNLOCK_LIST
にセット
562
リストロックフラグ
SET LIST LOCK FLAG 424 =
UNLOCK_LIST
にセット
570
失敗
FAIL
564
成功
SUCCEED
572

FIG. 5B

【図10】

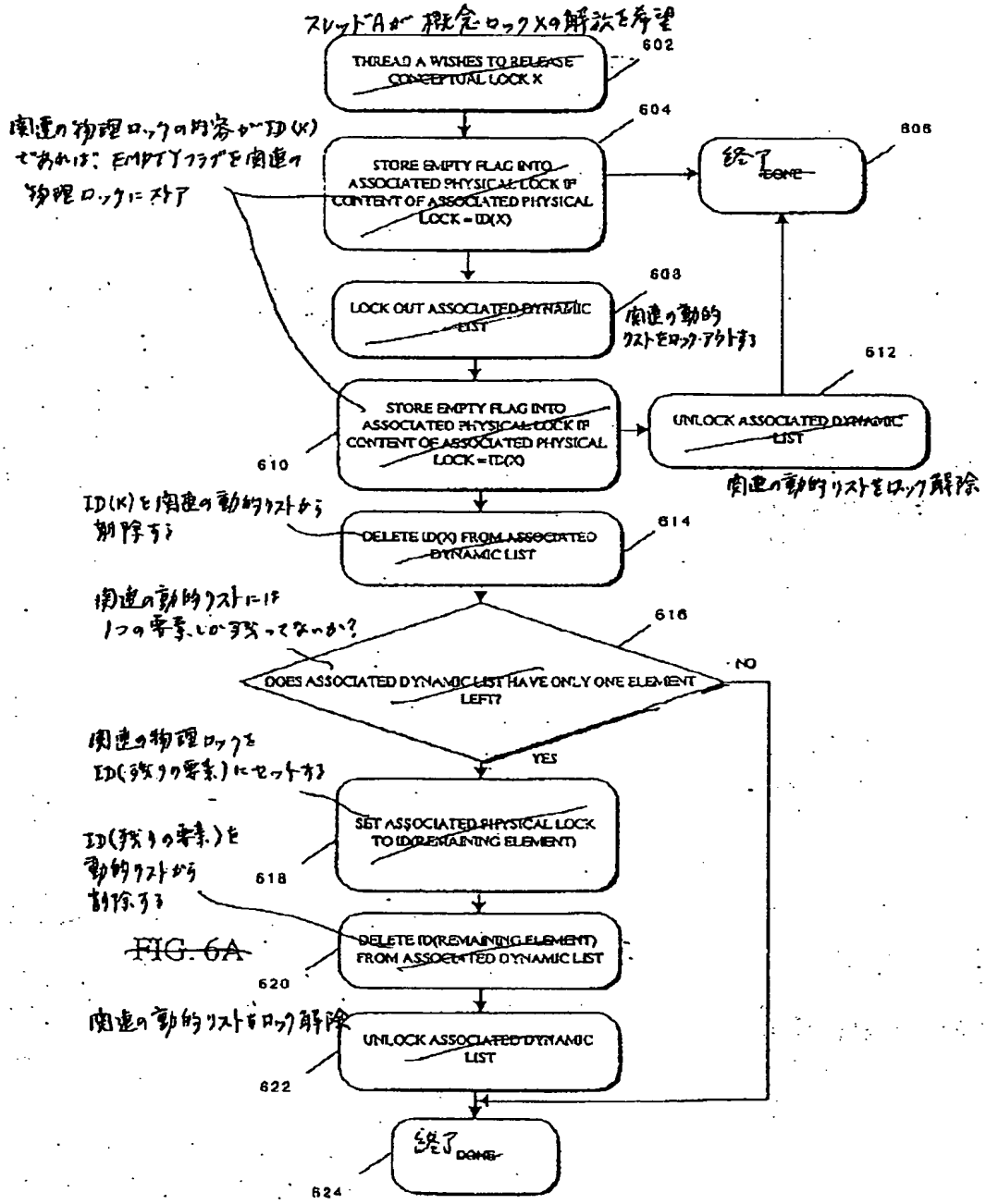

スレッドAが概念ロックXの解放を希望
THREAD A WISHES TO RELEASE CONCEPTUAL LOCK X
602
関連の物理ロックの内容がID(X)であれば、EMPTYフラグを関連の物理ロックに格納
STORE EMPTY FLAG INTO ASSOCIATED PHYSICAL LOCK IF CONTENT OF ASSOCIATED PHYSICAL LOCK = ID(X)
604
終了 DONE
606
LOCK OUT ASSOCIATED DYNAMIC LIST
608
関連の動的リストをロックアウトする
STORE EMPTY FLAG INTO ASSOCIATED PHYSICAL LOCK IF CONTENT OF ASSOCIATED PHYSICAL LOCK = ID(X)
610
UNLOCK ASSOCIATED DYNAMIC LIST
612
関連の動的リストをロック解除
ID(X)を関連の動的リストから削除する
DELETE ID(X) FROM ASSOCIATED DYNAMIC LIST
614
関連の動的リストには1つの要素しか残ってないか?
DOES ASSOCIATED DYNAMIC LIST HAVE ONLY ONE ELEMENT LEFT?
616
NO
YES
関連の物理ロックをID(残りの要素)にセットする
SET ASSOCIATED PHYSICAL LOCK TO ID(REMAINING ELEMENT)
618
ID(残りの要素)を動的リストから削除する
DELETE ID(REMAINING ELEMENT) FROM ASSOCIATED DYNAMIC LIST
620
関連の動的リストをロック解除
UNLOCK ASSOCIATED DYNAMIC LIST
622
終了 DONE
624

FIG. 6A

【図11】

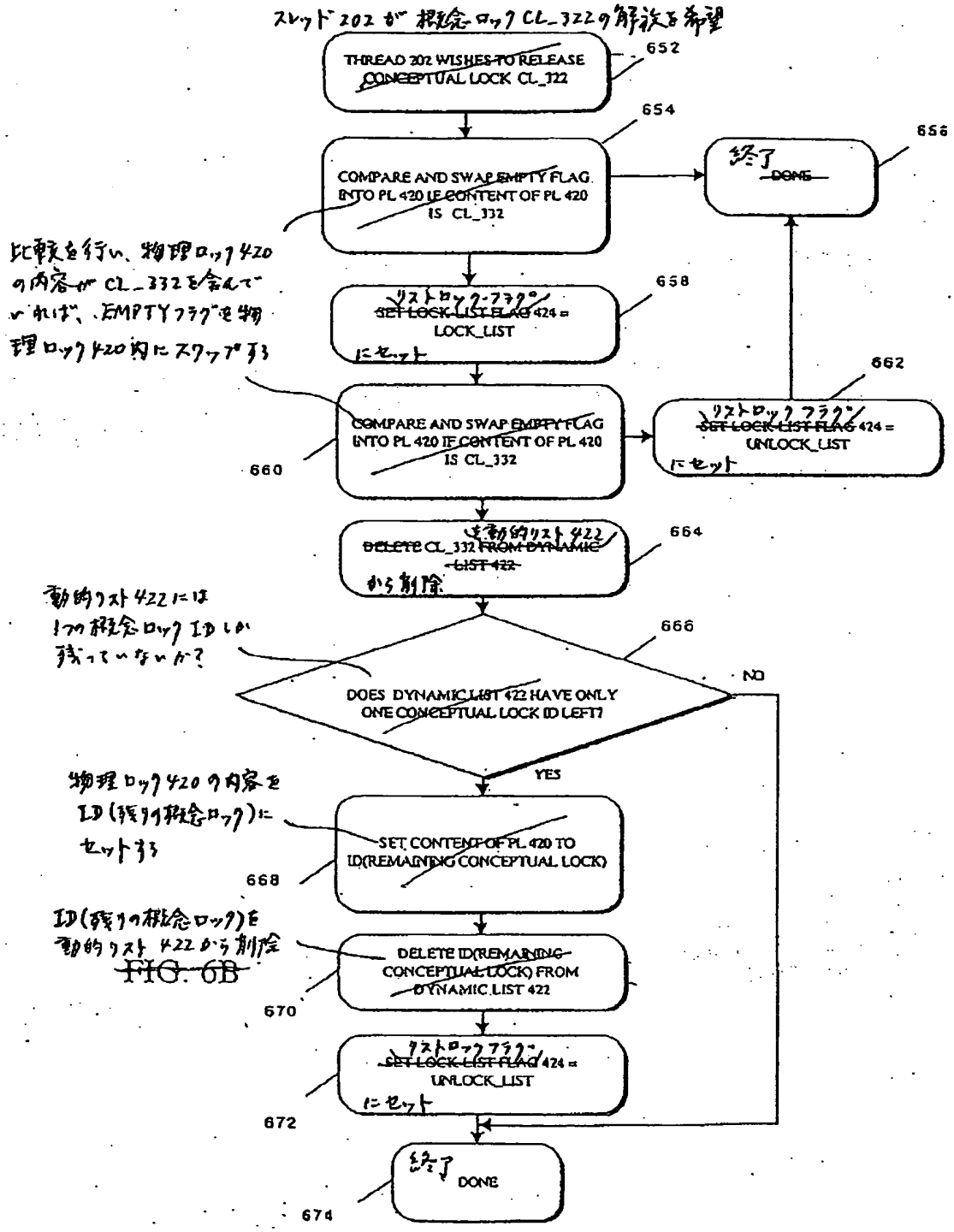

スレッド202 が 概念ロック CL_332の解放を希望
THREAD 202 WISHES TO RELEASE CONCEPTUAL LOCK CL_332
652
654
COMPARE AND SWAP EMPTY FLAG INTO PL 420 IF CONTENT OF PL 420 IS CL_332
656
終了 DONE
比較を行い、物理ロック420の内容がCL_332を含んでいれば、EMPTYフラグを物理ロック420内にスワップする
658
リストロックフラグ SET LOCK-LIST FLAG 424 = LOCK_LIST にセット
662
リストロックフラグ SET LOCK-LIST FLAG 424 = UNLOCK_LIST にセット
660
COMPARE AND SWAP EMPTY FLAG INTO PL 420 IF CONTENT OF PL 420 IS CL_332
664
動的リスト422 DELETE CL_332 FROM DYNAMIC LIST 422 から削除
動的リスト422には1つの概念ロックIDしか残っていないか?
666
DOES DYNAMIC LIST 422 HAVE ONLY ONE CONCEPTUAL LOCK ID LEFT?
NO
YES
物理ロック420の内容をID(残りの概念ロック)にセットする
668
SET CONTENT OF PL 420 TO ID(REMAINING CONCEPTUAL LOCK)
ID(残りの概念ロック)を動的リスト422から削除
670
DELETE ID(REMAINING CONCEPTUAL LOCK) FROM DYNAMIC LIST 422
672
リストロックフラグ SET LOCK-LIST FLAG 424 = UNLOCK_LIST にセット
終了 DONE
674

FIG. 6B

フロントページの続き

(71)出願人　591064003
901 SAN ANTONIO ROAD
PALO ALTO, CA 94303, U.
S. A.

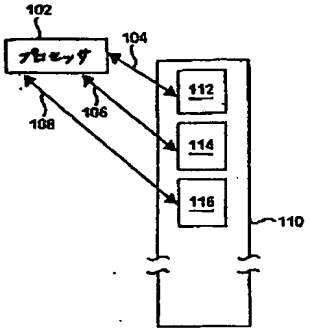
(72)発明者　マーク・ドナルド・ヒル
アメリカ合衆国・53705・ウィスコンシン
州・マディソン・チャンバーレイン　アヴ
ェニュ・2124

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１１年（１９９９）７月２日

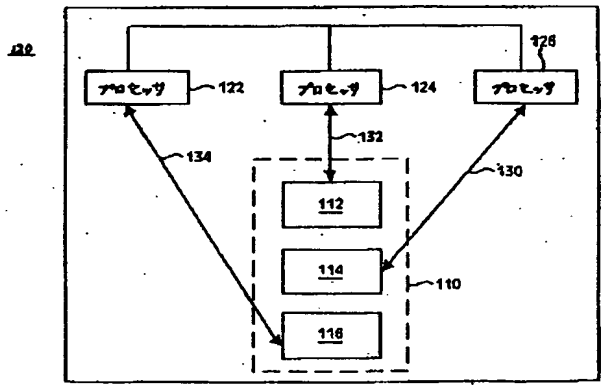
【公開番号】特開平１０－１８７５２７
【公開日】平成１０年（１９９８）７月２１日
【年通号数】公開特許公報１０－１８７６
【出願番号】特願平９－１８７２６９
【国際特許分類第６版】
G06F 12/00 572
9/46 340
【ＦＩ】
G06F 12/00 572 A
9/46 340 F

【手続補正書】
【提出日】平成１０年３月２３日
【手続補正１】
【補正対象書類名】図面
＊【補正対象項目名】全図
【補正方法】変更
＊【補正内容】

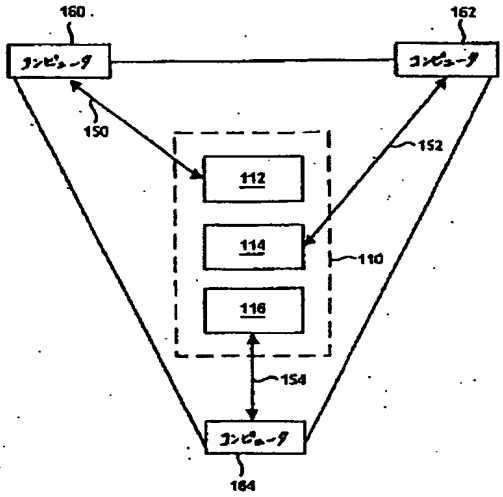
【図１】



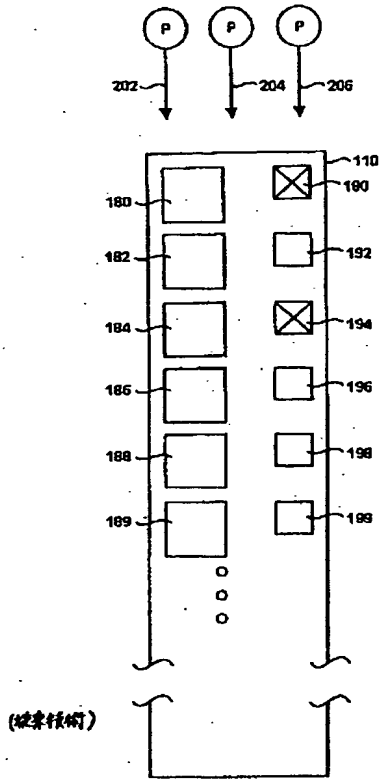
【図２】



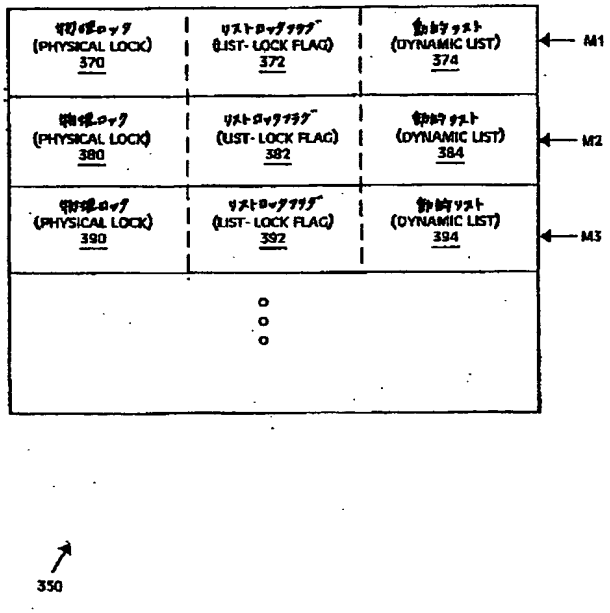
【図３】


160
162
コンピュータ
コンピュータ
150
152
112
114
110
116
154
コンピュータ
164

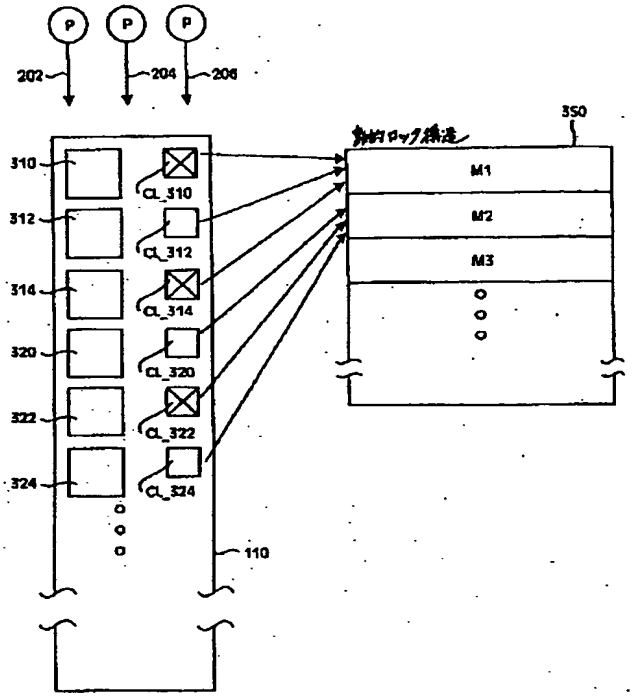
【図４】


P
P
P
202
204
206
110
180
180
182
192
184
194
186
196
188
198
189
199
(従来技術)

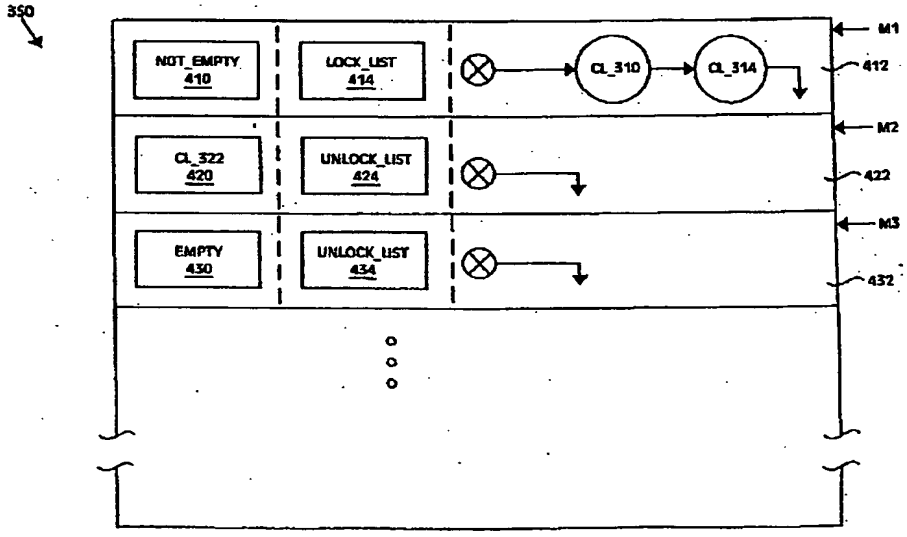
【図６】


(PHYSICAL LOCK)
370
(LIST- LOCK FLAG)
372
(DYNAMIC LIST)
374
M1
(PHYSICAL LOCK)
380
(LIST- LOCK FLAG)
382
(DYNAMIC LIST)
384
M2
(PHYSICAL LOCK)
390
(LIST- LOCK FLAG)
392
(DYNAMIC LIST)
394
M3
350

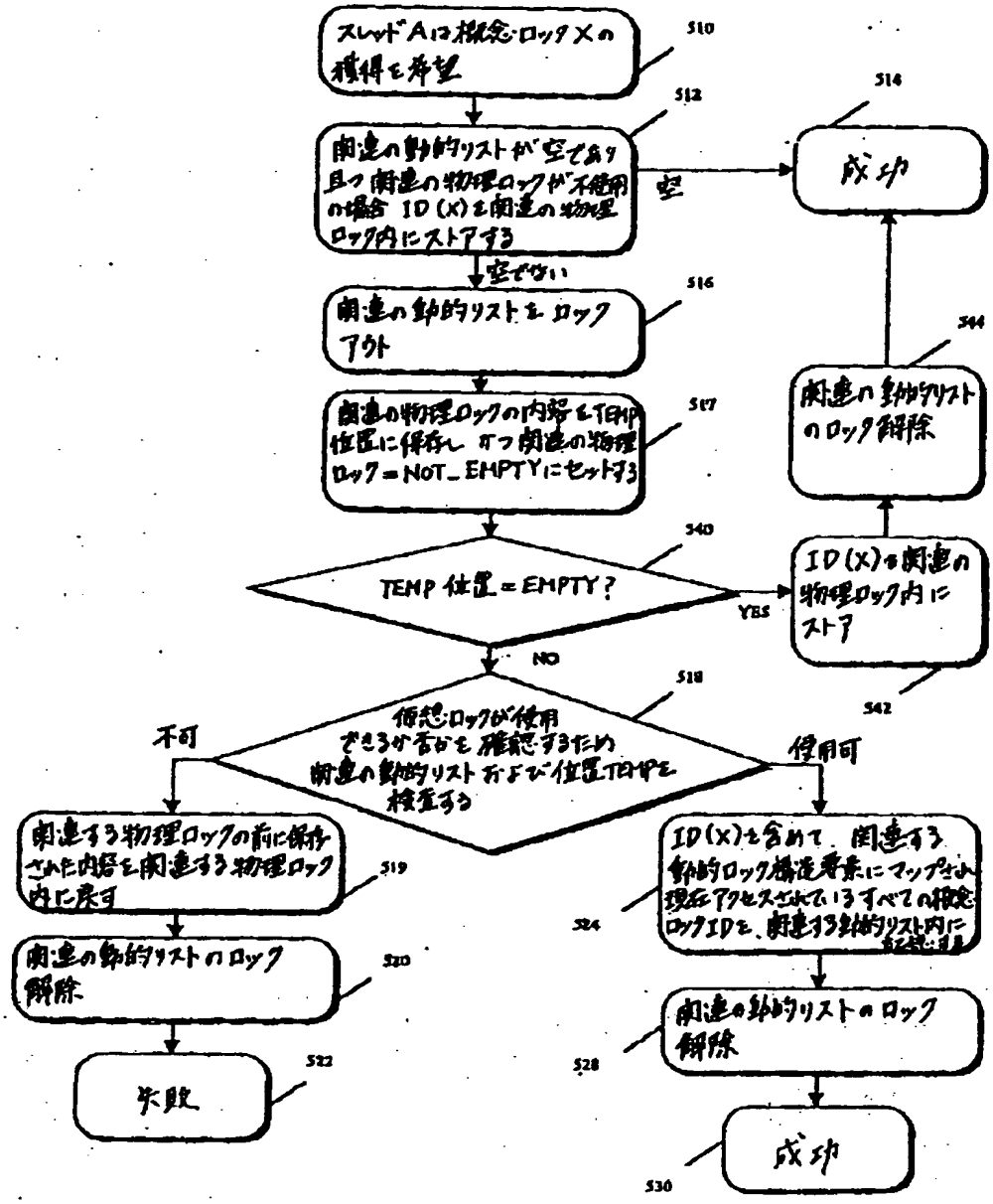
【図５】


P
P
P
202
204
206
350
M1
M2
M3
310
312
314
320
322
324
CL_310
CL_312
CL_314
CL_320
CL_322
CL_324
110

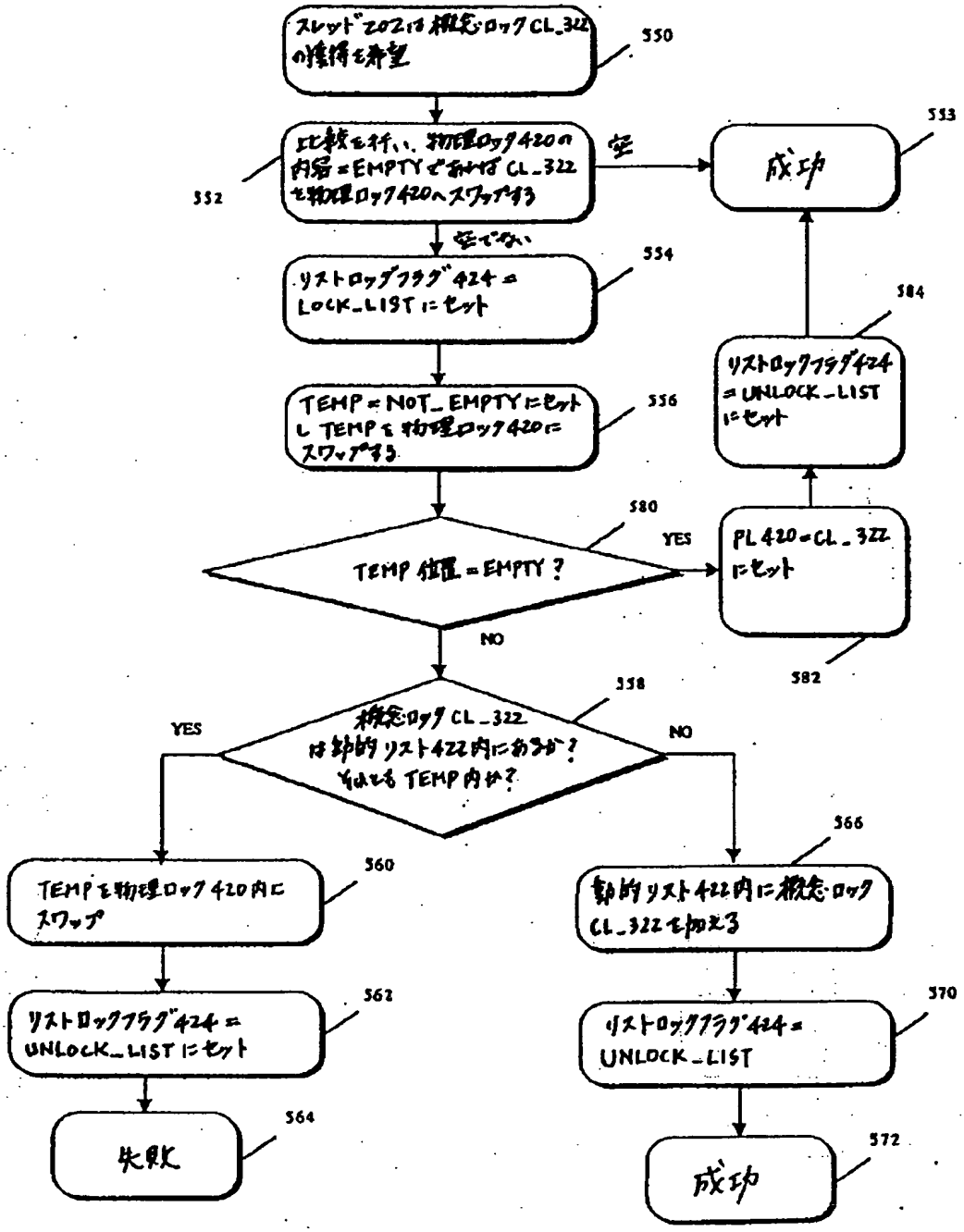
【図７】


350
NOT_EMPTY
410
LOCK_LIST
414
CL_310
CL_314
M1
412
M2
CL_322
420
UNLOCK_LIST
424
422
M3
EMPTY
430
UNLOCK_LIST
434
432

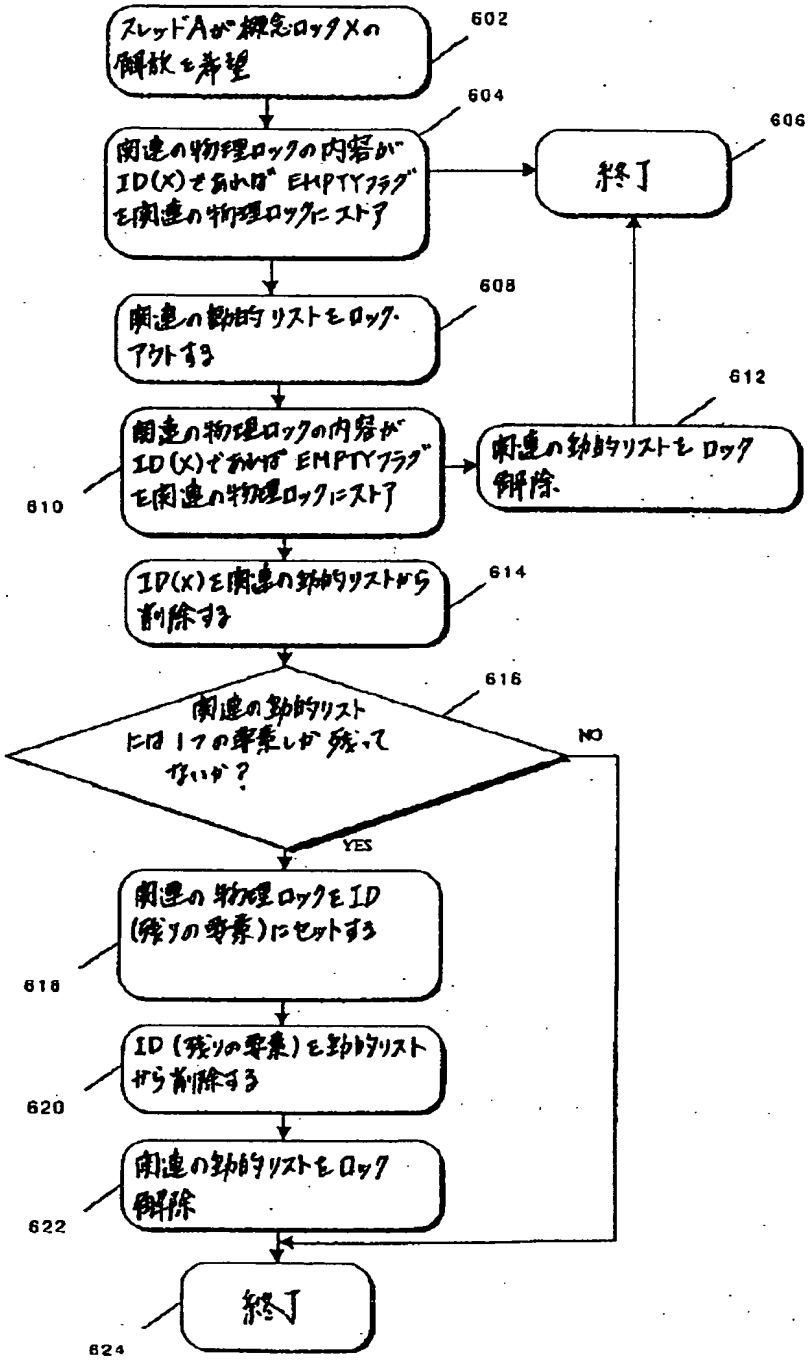
【図８】


スレッドAは概念ロックXの獲得を希望
510
関連の動的リストが空であり且つ関連の物理ロックが不使用の場合 ID(X)を関連の物理ロック内にストアする
512
空
成功
514
空でない
関連の動的リストをロックアウト
516
関連の物理ロックの内容をTEMP位置に保存し且つ関連の物理ロック＝NOT_EMPTYにセットする
517
関連の動的リストのロック解除
544
TEMP位置＝EMPTY?
540
YES
ID(X)を関連の物理ロック内にストア
542
NO
仮想ロックが使用できるか否かを確認するため関連の動的リスト及び位置TEMPを検査する
518
不可
使用可
関連する物理ロックの前に保存された内容を関連する物理ロック内に戻す
519
関連の動的リストのロック解除
520
失敗
522
ID(X)を含めて、関連する動的ロック構造要素にマップされ現在アクセスされているすべての概念ロックIDを、関連する動的リスト内に
524
関連の動的リストのロック解除
528
成功
530

【図９】


スレッド202は概念ロックCL_322の獲得を希望
550
比較を行い、物理ロック420の内容=EMPTYであればCL_322を物理ロック420へスワップする
552
空
成功
553
空でない
リストロックフラグ424=LOCK_LISTにセット
554
リストロックフラグ424=UNLOCK_LISTにセット
584
TEMP=NOT_EMPTYにセットしTEMPを物理ロック420にスワップする
556
TEMP値=EMPTY?
580
YES
PL420=CL_322にセット
582
NO
概念ロックCL_322は静的リスト422内にあるか?それともTEMP内か?
558
YES
NO
TEMPを物理ロック420内にスワップ
560
リストロックフラグ424=UNLOCK_LISTにセット
562
失敗
564
静的リスト422内に概念ロックCL_322を加える
566
リストロックフラグ424=UNLOCK_LIST
570
成功
572

【図１０】


スレッドAが排他ロックXの解放を希望
602
関連の物理ロックの内容がID(X)であればEMPTYフラグを関連の物理ロックにストア
604
終了
606
関連の動的リストをロック・アクトする
608
関連の物理ロックの内容がID(X)であればEMPTYフラグを関連の物理ロックにストア
610
関連の動的リストをロック解除
612
ID(X)を関連の動的リストから削除する
614
関連の動的リストには1つの要素しか残ってないか？
616
NO
YES
関連の物理ロックをID(残りの要素)にセットする
618
ID(残りの要素)を動的リストから削除する
620
関連の動的リストをロック解除
622
終了
624